

**IBM Cognos Framework Manager:
Design Metadata Models (v10.1)
Volume 2
Student Guide
Course Code: B5152**

*IBM Cognos BI Framework Manager: Design
Metadata Models (v10.1)*

B5152

ERC: 3.0

Published December 2010

Licensed Materials – Property of IBM

© Copyright IBM Corp. 2003, 2010

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM corp.

IBM, the IBM logo, ibm.com and Cognos are trademarks of international Business Machines Corp., registered in many jurisdictions worldwide.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

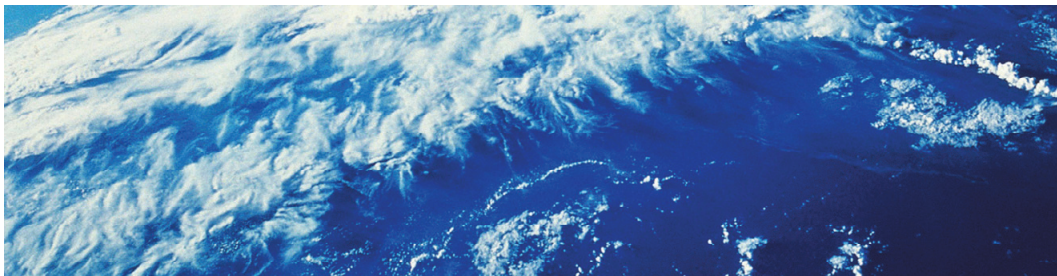
Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Other product and service names might be trademarks of IBM or other companies.



Create the Presentation View

IBM Cognos BI

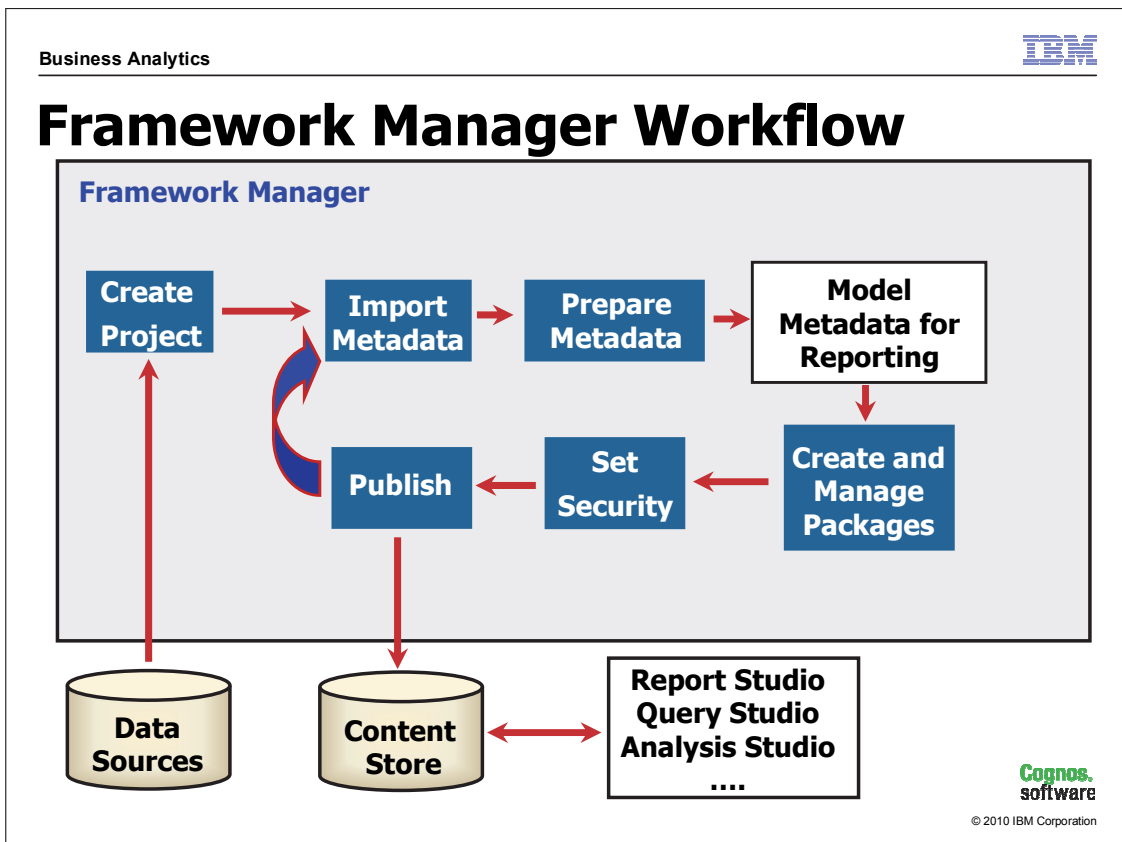


Business Analytics

© 2010 IBM Corporation

Objectives

- At the end of this module, you should be able to:
 - identify the dimensions associated with a fact table
 - identify conformed vs. non-conformed dimensions
 - create star schema groupings to provide authors with logical groupings of query subjects
 - rapidly create a model using the Model Design Accelerator

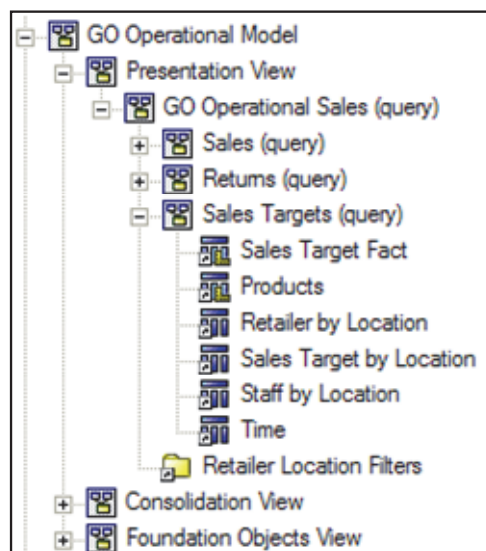


This module teaches how to create a presentation view consisting of logical groupings of query subjects (facts and their related dimensions) that focus on various areas of the business.

Create a Presentation View

Recommendation #10

- Provide a logical and simplified presentation of metadata and reporting tools for report authors.



Cognos.
software

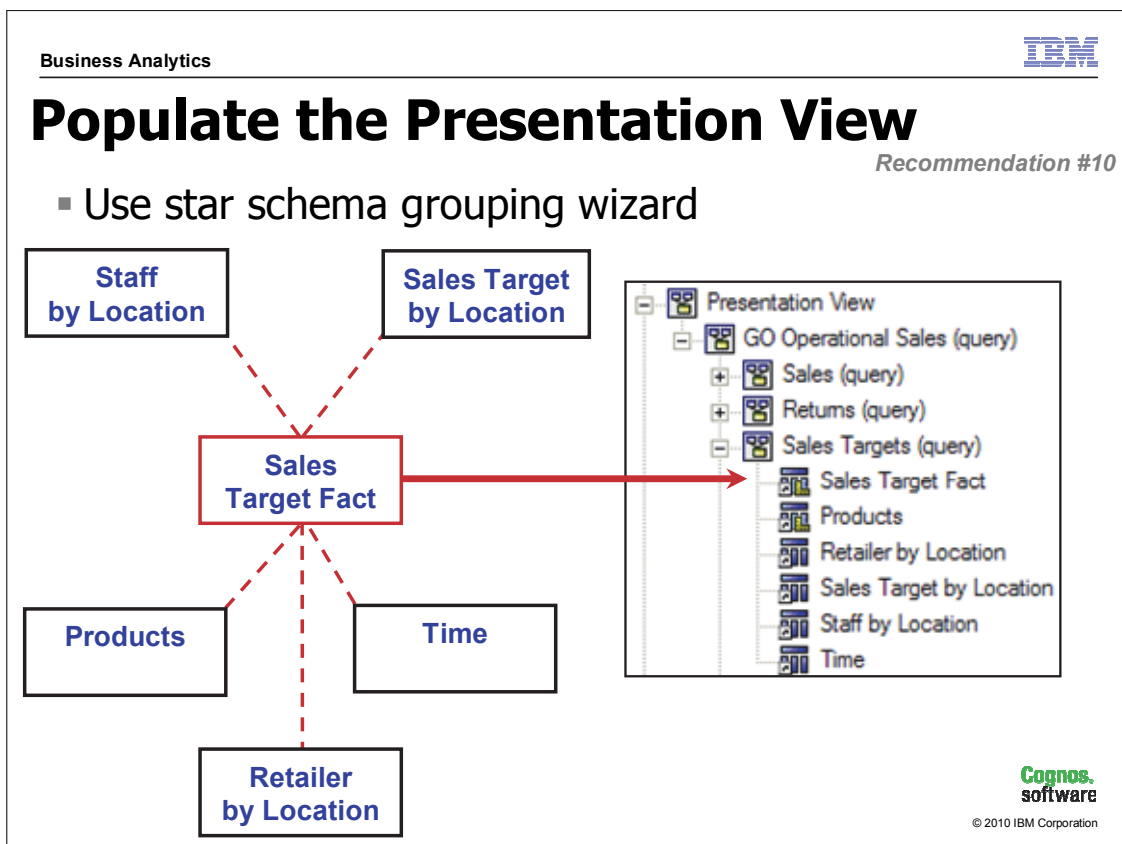
© 2010 IBM Corporation

A large part of your goal as a metadata modeler is to create a simplified view for report authors. This can be done by presenting the metadata in a logical manner and providing authors with commonly used tools such as filters or calculations.

The Presentation View in the slide example consists of shortcuts to Consolidation View model query subjects, arranged in star schema groupings (a fact query subject and all its related dimensions).

Generally, the Consolidation View and Foundation Objects View are hidden from report authors.

You do not have to model and present as a star schema. For example, if your model is designed to satisfy only a certain set of pre-built reports from which authors cannot stray, then you can model your metadata to that specific end. However, if you are modeling to a broader and largely ad hoc audience, then modeling as a star schema is an excellent choice for achieving predictable results.



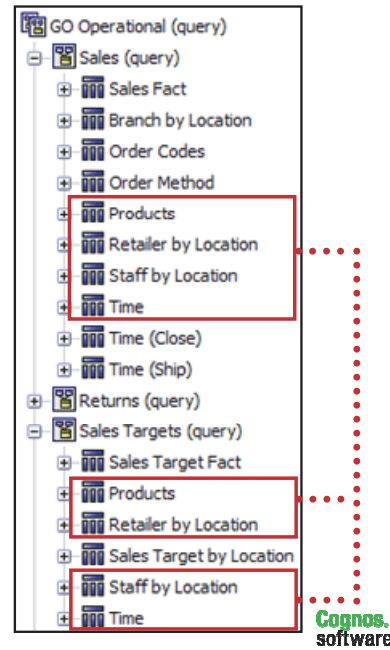
The Create Star Schema Grouping wizard creates logical groupings of central fact tables and their related dimensions. These groupings consist of shortcuts to the underlying objects and are placed in a namespace so that the same dimension names can occur in other star schema groupings. This allows authors to identify conformed dimensions.

As you model, you should document your logical groupings with a dimension map. You can then use the dimension map to quickly create your star schema groupings.

In the slide example, the objects on the left are model query subjects in the Consolidation View, which are based on objects in the Foundation Objects View. The model query subjects are related to each other in the Foundation Objects View (represented by the dashed lines in the diagram above). These objects are then grouped for presentation as shown on the right side of the diagram.

Identify Conformed Dimensions

- Based on matching names
- Must use at least one conformed dimension to report across facts to:
 - allow for stitch queries
 - ensure correct aggregation for each fact



© 2010 IBM Corporation

Modelers and authors can quickly identify conformed dimensions in the Presentation View based on naming conventions. If designed correctly, dimensions with the exact same name in different namespaces are shared between the facts.

Dimensions that are not shared between facts (non-conformed) can still be used in multi-fact queries providing at least one conformed dimension is used.

Demo 1: Create the Presentation View

Purpose:

To present report authors with an intuitive view of the metadata, you will create a view based on star schema groupings of your relational metadata. You will use the dimension map provided to you to easily create these groupings. You will then create a package specific to your new view, publish it, and test it.

Components: Framework Manager, Business Insight Advanced

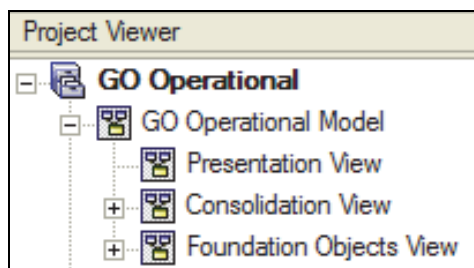
Project: GO Operational

Package: GO Operational (query)

Task 1. Use star schema groupings to populate the Presentation View.

1. In **Framework Manager**, close any projects that may be open, and then open the **GO Operational** project located at **C:\Edcognos\B5152\CBIFM-Start Files\Module 12\GO Operational**.
2. In the **Project Viewer** pane, under the **GO Operational Model** namespace, create a new namespace called **Presentation View**, and then drag it above the **Consolidation View** namespace.

The results appear as follows:



3. In the **Presentation View** namespace, create a new namespace called **GO Operational Sales (query)**.

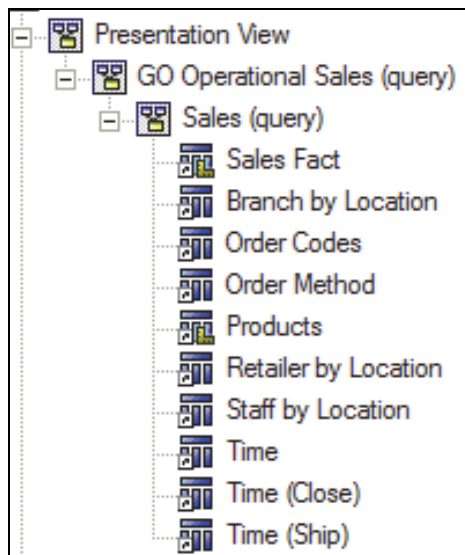
The (query) suffix in this case indicates that these objects are strictly relational items and not dimensionally modeled relational (DMR) objects. DMR objects will be created in a later module.

You will populate this new namespace with star schema groupings of your Consolidation View model query subjects.

4. In the **Consolidation View**, select **Sales Fact** and all its related dimensions identified in your **B5152-Requirements_handout** sheet.
5. Right-click on one of the selected objects, and then click **Create Star Schema Grouping**.
6. Change the **Namespace name** to **Sales (query)**, and then click **OK**.
7. Drag the new namespace into the **GO Operational Sales (query)** namespace in the **Presentation View**.

8. Expand **Sales (query)**.

The results appear as follows:



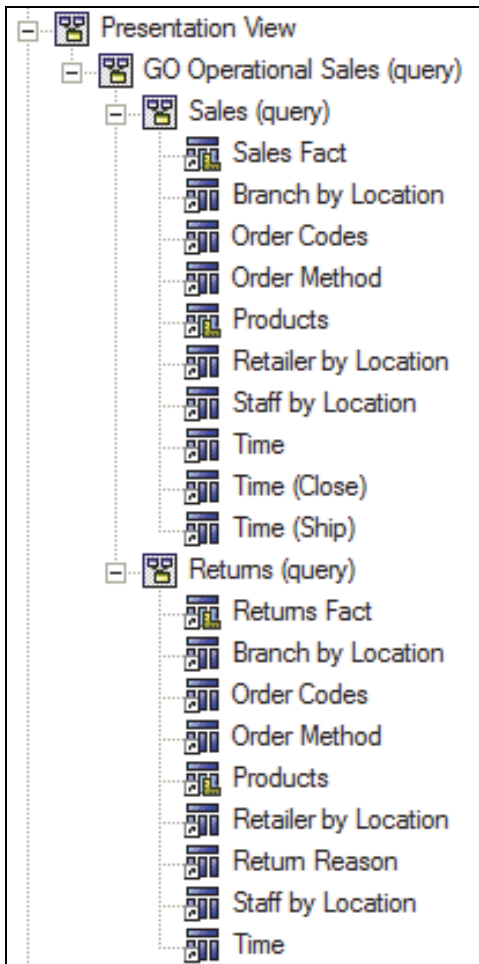
Note that all the model query subjects are shortcuts to the ones in the Consolidation View. You have simply grouped the ones needed for sales queries in one place.

If the dimensions are not listed alphabetically, you can use the Reorder feature to sort them and then drag Sales Fact to the top for easy access.

9. Repeat steps 4 to 8 for the following fact query subject:

- **Returns Fact** - Namespace name = **Returns (query)**

The results appear as follows:



The conformed dimensions are clearly visible (such as Products and Staff by Location). This allows authors to create queries across sales and returns facts.

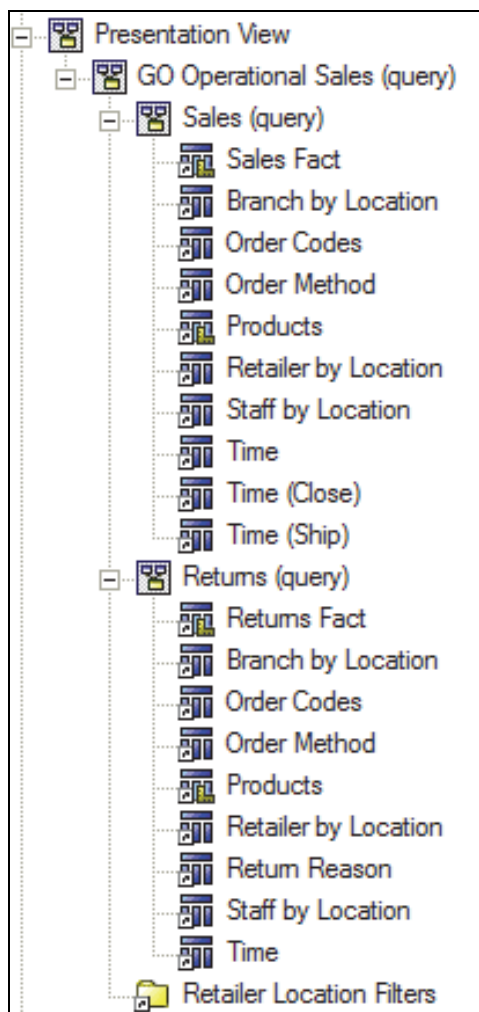
Important note: If you encounter any problems with these groupings during your testing, you will need to trace back to the Foundation Objects View to ensure all the proper relationships are in place and that there are no unresolved reporting traps.

Task 2. Make model filters available to report authors.

You will now make the Retailer Location Filters available in the Presentation View by using a shortcut.

1. In the **Consolidation View**, expand **Model Filters**, right-click **Retailer Location Filters**, point to **Create**, and click **Shortcut**.
2. Move the shortcut to the **GO Operational Sales (query)** namespace, and then rename it to **Retailer Location Filters**.

The results appear as follows:

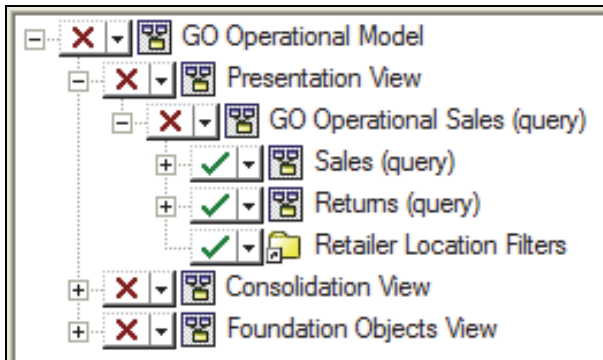


3. Save the project.

Task 3. Create and publish a Presentation View package.

You will create a new package containing just the metadata that was modeled for report authors.

1. Right-click **Packages**, point to **Create**, and then click **Package**.
2. In the **Name** box, type **GO Operational (query)**, and then click **Next**.
3. Clear **GO Operational Model**, expand **Presentation View>GO Operational Sales (query)**.
4. Select all children of **GO Operational Sales (query)** as shown below:



5. Click **Finish**.

You are prompted to open the Publish Package wizard.

6. Click **Yes**.
7. Clear the **Enable model versioning** check box, click **Next** twice, and then click **Publish**.

Tip: You can open IBM Cognos Connection from this dialog in order to quickly view and test your work.

8. Click **Yes**, and then click **Finish**.

The Verify Model dialog box appears listing informational messages that indicate underlying objects will be published with the package but hidden from authors. This is necessary as IBM Cognos will require information from these items to properly generate queries.

9. Click **Close**, and then save and close the project.

Results:

By using star schema groupings to populate the Presentation View, you have created an easy-to-understand view of the metadata that follows two simple rules:

1. When writing queries, use associated dimensions and facts (these are logically grouped using namespaces).

2. When writing multi-fact queries, use at least one conformed dimension, which can be identified by naming conventions.

Workshop 1: Create the Presentation View

Using the Star Schema Grouping wizard, you will create a logical grouping for sales target information.

To accomplish this, you will:

- Select Sales Target fact in the Consolidation View and all its related dimensions (use the dimension map provided)
- Use the Star Schema Grouping wizard to create a new namespace called Sales Targets (query) containing the selected items
- Move the new namespace to the GO Operational Sales (query) namespace below Returns (query)
- Update the GO Operational (query) package to include the new namespace
- Publish the package and test the Sales Target (query) in Query Studio with the following items:
 - Sales Target by Location>Sales Target Country
 - Sales Target Fact>Sales Target

An error message is returned. Return to the Project to resolve the issue and then close the browser to clear the cache memory before testing. Retest the package in Query Studio.

For more detailed information outlined as tasks, see the Task Table on the next page.

To see the desired filters, see the Workshop Results section that follows the Task Table.

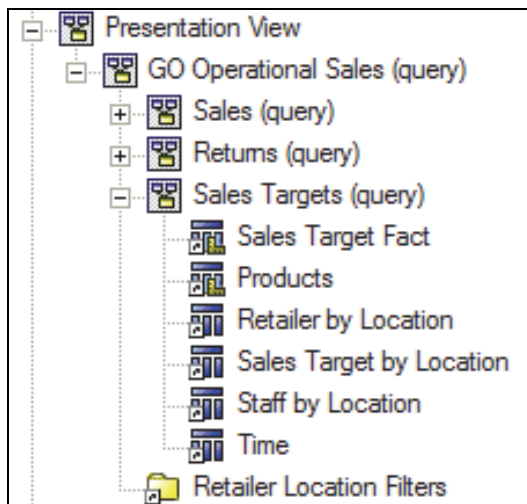
Workshop 1: Task Table

Task	Where to Work	Hints
1. Create star schema grouping for sales targets.	Project Viewer, Star Schema Grouping wizard	<ul style="list-style-type: none"> • In the Consolidation View, select Sales Target Fact and all related dimensions identified in your dimension map handout • Call the new namespace Sales Target (query) • Move the new namespace to the GO Operational (query) located in the Presentation View namespace and place below Returns (query)
2. Update GO Operational (query) package and publish.	Project Viewer, Package Definition	<ul style="list-style-type: none"> • Open the GO Operational (query) package definition • Select the Sales Targets(query) namespace under GO Operational (query) • Publish the package

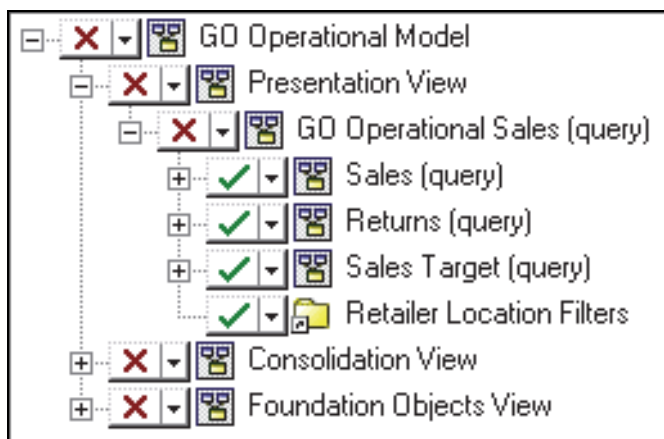
Task	Where to Work	Hints
3. Test Sales Targets (query) objects in Query Studio.	IBM Cognos Connection, Query Studio	<ul style="list-style-type: none"> • Launch IBM Cognos Connection • Launch Query Studio selecting the GO Operational (query) package • Add the following items to the report: <ul style="list-style-type: none"> • Sales Target by Location>Sales Target Country • Sales Target Fact>Sales Target • Read the error message and then return to Framework Manager and investigate in the Foundation Objects View (working your way backwards from the Consolidation View) • Recreate missing relationship, publish package again, and then test in Query studio • Close the browser, without saving and close with saving changes in Framework Manager

Workshop 1: Workshop Results

Once you have created your Sales Targets (query) star schema grouping, the Presentation View should appear as shown below:

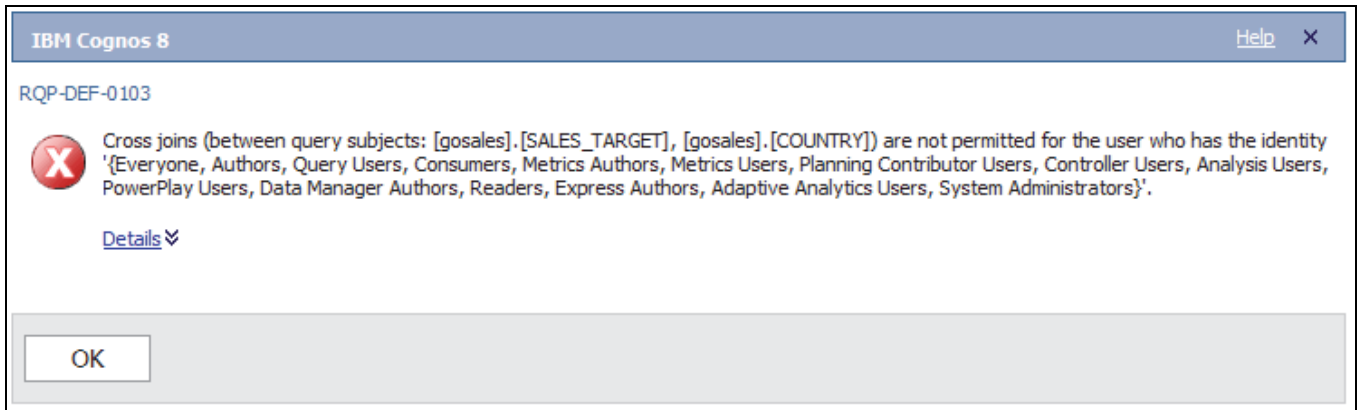


Once you have updated the GO Operational (query) package definition it should appear as shown below:



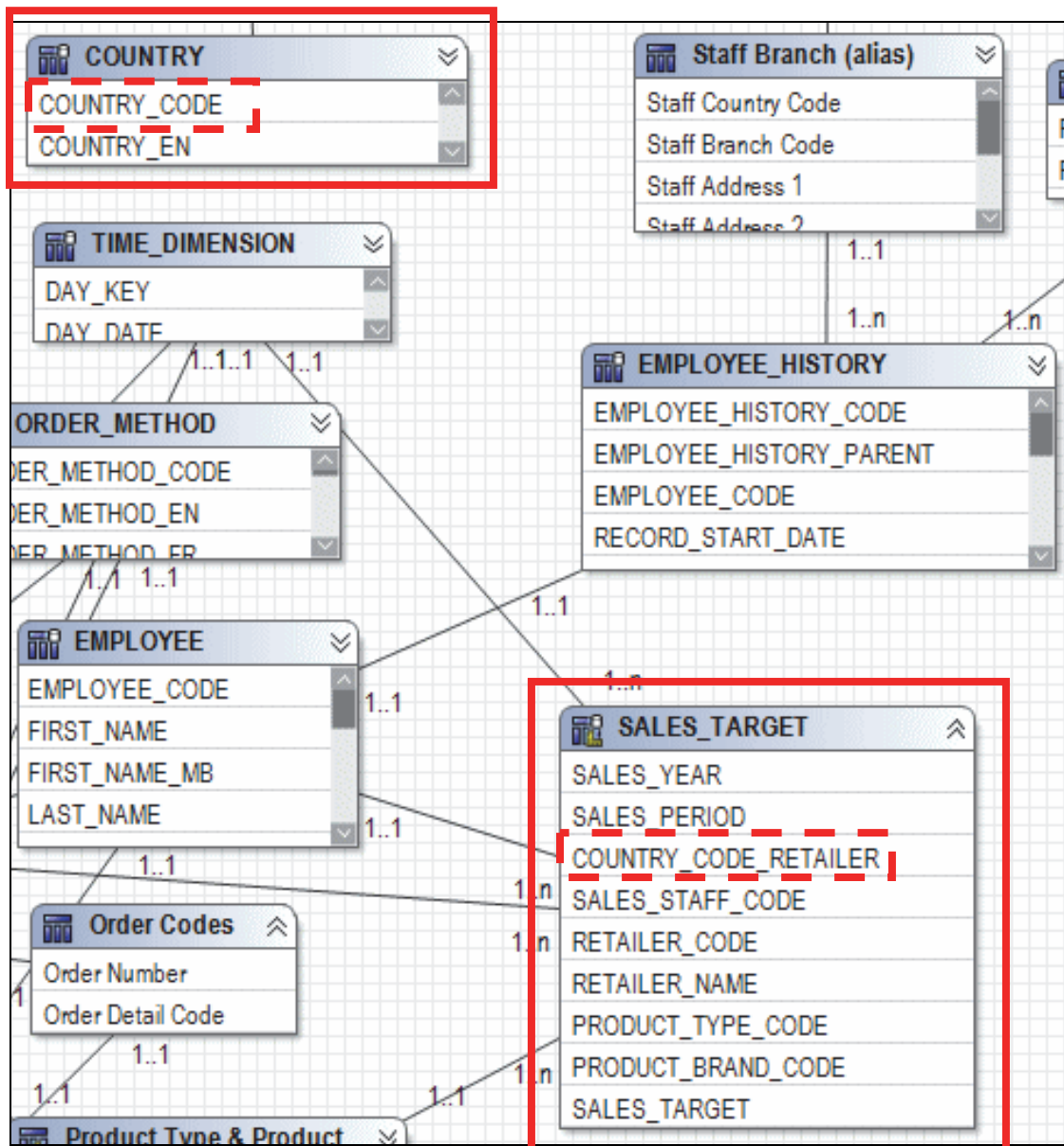
Workshop 1: Workshop Results

After testing the Sales Targets (query) items, your error message should appear as shown below:



Workshop 1: Workshop Results

After investigating the Foundation Objects View (working you way backwards from the Consolidation View), you should see a missing relationship in the Diagram between COUNTRY and SALES_TARGET as shown below:



The relationship was accidentally deleted during the modeling process.

Workshop 1: Workshop Results

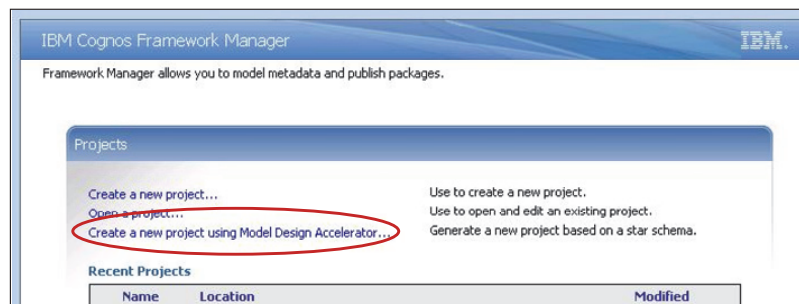
After you recreate the relationship, publish the package and test in Query Studio, your report should appear as shown below:

Sales Target Country	Sales Target
Australia	98,545,000
Austria	128,744,500
Belgium	101,979,100
Brazil	123,728,300
Canada	272,116,900
China	286,772,000
Denmark	55,215,000
Finland	169,332,500
France	257,675,400
Germany	235,055,620
Italy	167,696,700
Japan	318,688,170
Korea	180,729,100
Mexico	149,668,200
Netherlands	165,116,400
Singapore	177,137,700
Spain	149,005,900
Sweden	86,559,000
Switzerland	90,307,000

The report works and returns results as expected.

Model Design Accelerator

- The Model Design Accelerator is a graphical utility designed to guide modelers through a simplified modeling process.



The Model Design Accelerator is a graphical utility designed to guide both novice and experienced modelers through a simplified modeling process. The Model Design Accelerator applies IBM Cognos best practices to quickly produce single star schemas.

Multiple star schemas can be created using the Model Design Accelerator several times and linking the results together. Additional features can be added to the model using standard Framework Manager functionality.

Demo 2: Rapidly Create a Model using the Model Design Accelerator

Purpose:

A senior manager wants to create reports about returned products to review returns data by product, customer, and return reason. Since this project has a limited scale, you will use the Model Design Accelerator to quickly produce a package for reporting.

A data warehouse has been created and will be used as it is better suited for reporting and ease of modeling.

Component: **Framework Manager**

Project: **GO Returns**

Task 1. Start the Model Design Accelerator and import data.

1. In **Framework Manager**, close any projects that may be open.
2. Click **Create a new project using Model Design Accelerator**.
The New Project dialog opens.
3. Navigate to **C:\Edcognos\B5152\Course_Project** in the **Location** box, and then click **OK**.
4. In the **Project name** box, type **GO Returns**.
The GO Returns folder is created by default and appears by default in the Location box.
5. Click **OK**.
The Select Languages dialog box appears. Set the default and design language for this project as English.
6. Ensure that **English** is selected, and then click **OK**.
The Metadata Wizard appears.

7. Select the **great_outdoors_warehouse** data source.
8. In the list of objects, expand **GOSALESDW>Tables**.
9. Select the following tables:

DIST_RETURNED_ITEMS_FACT
DIST_RETURN_REASON_DIM
SLS_PRODUCT_DIM
SLS_PRODUCT_LINE_LOOKUP
SLS_PRODUCT_LOOKUP
SLS_PRODUCT_TYPE_LOOKUP
SLS_RTL_DIM

10. Click **Continue**.

The IBM Cognos Framework Manager User Guide window opens, displaying information about the Model Design Accelerator.

The information in this window explains the steps to create a model using the Model Design Accelerator.

11. Close the IBM Cognos Framework Manager User Guide.

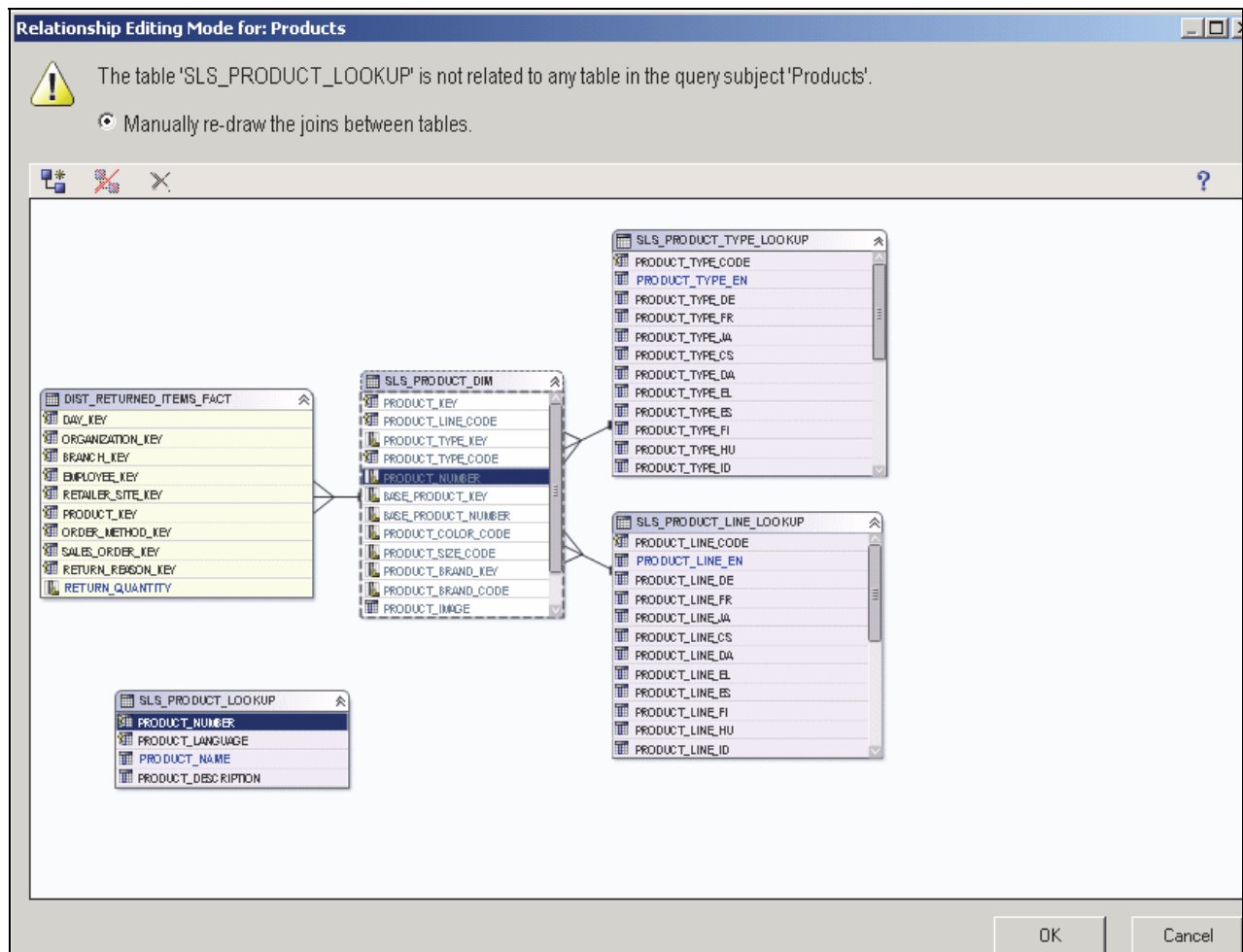
Task 2. Create a Returns fact table.

1. In the **Model Accelerator** pane, right-click the **Fact Query Subject** in the center of the pane and click **Rename**.
2. Type **Returns Fact** to rename the fact query subject.
3. Press **Enter**.
4. In the **Explorer Tree** pane, expand the **DIST_RETURNED_ITEMS_FACT** table.
5. Drag the **RETURN_QUANTITY** data item into the **Returns Fact** query subject.

Task 3. Create a Product Dimension Table.


1. Rename **New Query Subject 1** to **Products**.
2. In the **Explorer Tree** pane, expand the **SLS_PRODUCT_LINE_LOOKUP** table.
3. Drag the **PRODUCT_LINE_EN** data item into the **Products** query subject.
4. Expand the **SLS_PRODUCT_TYPE_LOOKUP** table.
5. Drag the **PRODUCT_TYPE_EN** data item into the **Products** query subject.
6. Expand the **SLS_PRODUCT_LOOKUP** table.

7. Drag the **PRODUCT_NAME** data item into the **Products** query subject.
The Relationship Editing Mode for: Products dialog opens.



This dialog opens because Framework Manager cannot determine the relationship between the SLS_PRODUCT_LOOKUP table and the DIST_RETURNED_ITEMS_FACT table. You will need to establish the relationship yourself.

8. Ctrl-click **SLS_PRODUCT_LOOKUP>PRODUCT_NUMBER** and **SLS_PRODUCT_DIM>PRODUCT_NUMBER**.

9. In the **Relationship Editing Mode** window, click **Create a Model Relationship between these Columns**  in the top left corner of the dialog.

The Modify the Relationship dialog opens.

10. From the **Relationship Cardinality** drop-down list, select **One to Many**.

The SLS_PRODUCT_LOOKUP table has an entry for each product for each language. This results in a one-to-many relationship with the PRODUCT table. Once you finish generating the basic model, you will add a filter to filter out all non-English product names, thus creating a one-to-one relationship.

11. Click **OK**, and then click **OK** again.

Task 4. Create a Retailer Dimension Table.

1. Rename **New Query Subject 2** to **Retailers**.
2. In the **Explorer tree** pane, expand the **SLS_RTL_DIM** table.
3. Drag the **RETAILER_TYPE_EN** and **RETAILER_NAME** data items into the **Retailers** query subject.
4. Double-click the **Retailers** table.
5. Double-click the link between the **SLS_RTL_DIM** and **DIST_RETURNED_ITEMS_FACT** tables.

Notice how the link between the tables is based on RETAILER_SITE_KEY. The Model Design Accelerator creates this join for you.

6. Click **Close**, and then close the **Query Subject Diagram: Retailers** window.

Task 5. Create a Return Reason Dimension Table.

1. Rename **New Query Subject 3** to **Return Reason**.
2. In the **Explorer tree** pane, expand the **DIST_RETURN_REASON_DIM** table.
3. Drag the **REASON_DESCRIPTION_EN** data item into the **Return Reason** query subject.
4. Right-click **New Query Subject 4** and press **Delete**.
5. Click **Generate Model**, then click **Yes**.

The Model Design Accelerator creates a model for you based on your selections.

Task 6. Add a filter to the model.

1. Expand the **Model** namespace in the Project Viewer.
2. Expand the **Physical View** namespace.
3. Expand the **GOSALESDW** namespace.
4. Double-click the **SLS_PRODUCT_LOOKUP** table.

The Query Subject Definition window opens.

5. Click the **Filters** tab, and then click **Add**.
6. Type **Language Filter** in the **Name** text box.
7. Drag the **PRODUCT_LANGUAGE** data item from the **Available Components** pane to the **Expression** box.
8. After **PRODUCT_LANGUAGE** type **= 'EN'**.
9. Click **OK**, click the **Test** tab, and then click **Test Sample** in the bottom right of the window.

All the values in the **PRODUCT_LANGUAGE** column should read "EN."


10. Click **OK**.

Task 7. Create a GO Returns package.

1. In the **Project Viewer**, right-click **Packages**, point to **Create** and then click **Package**.
2. Type **GO Returns** in the **Name** text box.
3. Click **Next**.
4. Click the green check mark beside the **Model** namespace.
All the green check marks turn to red x's.
5. Expand the **Presentation View** namespace.
6. Change the red x beside **Returns Fact**, **Products**, **Retailers**, and **Return Reason** tables to green check marks.
7. Click **Finish**.
8. Click **Yes**.
9. Clear the **Enable model versioning** check box, accept the remaining defaults, and then click **Next**.
10. On the **Add Security** page, click **Next**.
11. Clear the **Verify the package before publishing** check box, and then click **Publish**.
12. Click **Finish** to close the wizard, and then save the project.

Task 8. Test the GO Returns package.

1. Log in to **IBM Cognos Connection** using username **admin** and password **Education1!**.
2. Click **Author Business Reports**.
3. Click the **GO Returns** package.

4. Click **Create new**, and then double-click **List**.
5. Expand the **Retailers** query subject.
6. Drag the **RETAILER_NAME** data item into the workspace.
7. Expand the **Return Reason** query subject.
8. Drag the **REASON_DESCRIPTION_EN** data item into the workspace to the right of **RETAILER_NAME**.
9. Expand the **Products** query subject.
10. Drag the **PRODUCT_LINE_EN** data item into the workspace to the right of **REASON_DESCRIPTION_EN**.
11. Expand the **Returns Fact** query subject.
12. Drag **RETURN_QUANTITY** into the workspace to the right of **REASON_DESCRIPTION_EN**.
13. Select the **RETAILER_NAME** column header in the workspace.
14. Click the Group  button in the toolbar.
15. Close all browser windows, save the project and then close Framework Manager.

Results:

The senior manager can now review returns data by return reason, retailer, and product. The model and package were created in much less time than it would have taken had you not used the Model Design Accelerator.

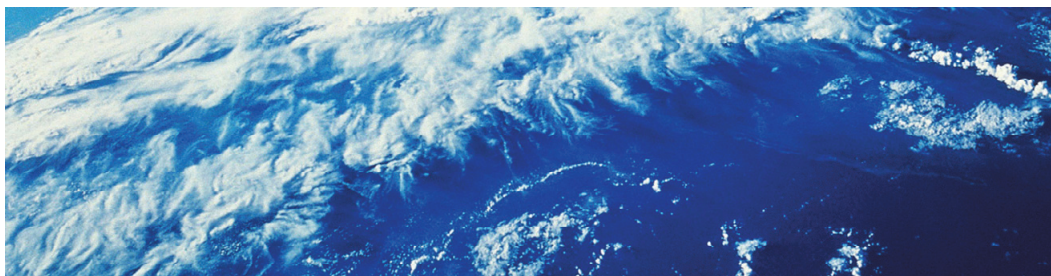
Summary

- You should now be able to:
 - identify the dimensions associated with a fact table
 - identify conformed vs. non-conformed dimensions
 - create star schema groupings to provide authors with logical groupings of query subjects
 - rapidly create a model using the Model Design Accelerator



Work with Different Query Subject Types

IBM Cognos BI



Business Analytics

© 2010 IBM Corporation

Objectives

- At the end of this module, you should be able to:
 - identify key differences and make recommendations for using data source, model, and stored procedure query subjects
 - identify the effects on generated SQL when modifying query subjects, SQL settings and relationships


Compare Query Subject Types

- Three types of query subjects:
 - data source
 - model
 - stored procedure
- Each can be foundation model objects used to create:
 - regular and measure dimensions
 - various business views

Data source query subject - maps to a corresponding object in the data source and uses a modifiable SQL statement to retrieve the data

Model query subject - maps to existing metadata in the model

Stored procedure query subject - executes a database stored procedure to retrieve or update data

Business Analytics


Data Source Query Subjects

- Data source query subjects are:
 - generated from an SQL statement
 - generally a simple, all-inclusive SQL statement
 - modifiable

SQL statement after import


Select * from [GOSL].RETURNED_ITEM

➔

Modified SQL statement

```

Select
  ORDER_HEADER.RETAILER_SITE_CODE,
  ORDER_HEADER.RETAILER_CONTACT_CODE,
  ORDER_HEADER.SALES_STAFF_CODE,
  ORDER_HEADER.SALES_BRANCH_CODE,
  ORDER_HEADER.ORDER_METHOD_CODE,
  ORDER_DETAILS.PRODUCT_NUMBER,
  RETURNED_ITEM.RETURN_CODE,
  RETURNED_ITEM.RETURN_DATE,
  RETURNED_ITEM.ORDER_DETAIL_CODE,
  RETURNED_ITEM.RETURN_REASON_CODE,
  RETURNED_ITEM.RETURN_QUANTITY
from
  [GOSL].RETURNED_ITEM,
  [GOSL].ORDER_HEADER ORDER_HEADER,
  [GOSL].ORDER_DETAILS ORDER_DETAILS
where
  RETURNED_ITEM.ORDER_DETAIL_CODE =
  ORDER_DETAILS.ORDER_DETAIL_CODE
and
  ORDER_HEADER.ORDER_NUMBER =
  ORDER_DETAILS.ORDER_NUMBER
          
```



© 2010 IBM Corporation

The SQL that is found on the SQL tab of a Query Subject Definition dialog populates the query item list, defines the scope of the query subject, and generates runtime SQL.

You can modify the SQL as required, to generate SQL that meets specific needs. You can also implement parameter driven dynamic SQL.

Alter the simple select statements as little as possible to generate the most efficient SQL and simplify model maintenance.

In the slide example we see the RETURNED_ITEM data source query subject SQL after import. It is a simple, all-inclusive select statement. If you do not alter this SQL and new columns are added to the table, they will be automatically reflected in Framework Manager when you update the query subject or test it. If you modify the SQL as seen on the right side of the slide example, new columns will not be reflected, and will need to be added manually in the SQL statement. As stated, sometimes customized SQL is required for a specific application.

Set SQL Type

- You can set the SQL type for data source query subjects.
- SQL Type setting includes:
 - Cognos
 - Native
 - Pass-Through

SQL settings can be specified from the Options link inside the Query Subject Definition dialog box or by enclosing your select statement in `{ }` for native SQL and `{ { } }` for pass-through SQL.

This setting is local to the query subject and impacts how a table-based query is defined and used in query generation.

SQL Type: Cognos SQL

- Adheres to SQL standards
- Can contain metadata from multiple data sources
- Has fewer database restrictions
- Works with all relational and tabular data sources
- Is portable

If you need to port your model from one vendor to another, use Cognos SQL since it works with all relational and tabular data sources. It also allows IBM Cognos to generate the most optimized SQL possible, for example by removing unused elements at query time.

SQL Type: Native SQL

- Allows SQL that is specific to your database
- May not be portable
- Cannot contain metadata from multiple data sources

When viewing generated Cognos SQL at run-time for a query subject that is set to Native SQL, the native SQL appears as a sub-query contained between {}.

SQL Type: Pass-Through SQL

- Use Pass-Through SQL when a database vendor does not extend support for a particular construct in a sub-query.
- IBM Cognos will pass anything you type directly to the database.

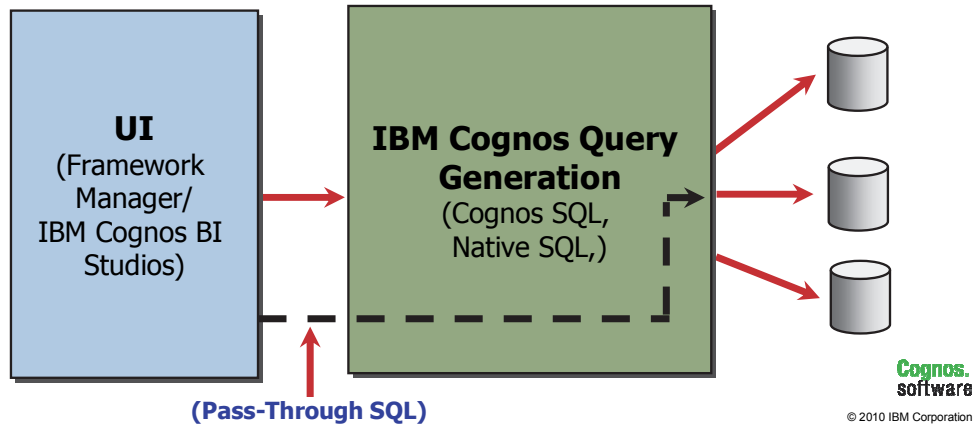
Pass-Through SQL lets you use native SQL without any of the restrictions the data source imposes on sub-queries.

With Cognos SQL and native SQL, when SQL is generated, IBM Cognos may create wrappers for certain queries that go around a sub-query construct and pass the entire construct (wrapper and sub-query) to the database. Some vendors may not support this. Pass-Through SQL will tell IBM Cognos to send only the sub-query to the database and then process the remaining SQL construct (wrapper) locally.

When viewing Cognos SQL for a query subject that is set to Pass-Through SQL, the native SQL that you typed will appear as a sub-query contained between `{ }`.

IBM Cognos Query Generation Architecture

- The IBM Cognos user interfaces submit SQL.
- The SQL option you have chosen will determine how IBM Cognos generates SQL.



Demo 1: Explore Generated SQL and Configure SQL Type Setting

Purpose:

As a metadata modeler, you should distinguish between SQL used to create and define data source query subjects and SQL generated by query subjects.

Not only will you explore this SQL, you have been asked to provide a query item that returns the current year for use in various reports. To accomplish this, you will use select construct and vendor specific function that requires you to change the SQL Type setting.

Component: Framework Manager

Project: GO Operational

Task 1. View data source query subject SQL.

1. In **Framework Manager**, close any projects that may be open, and then open the **GO Operational** project located at **C:\Edcognos\B5152\CBIFM-Start Files\Module 13\GO Operational**.
2. If prompted, log in as User ID **admin**, and Password **Education1!**.
3. In the **Project Viewer** pane, expand **GO Operational Model>Foundation Objects View>gosales**.
4. Double-click **SALES_TARGET**.

On the SQL tab, notice the simple, all-inclusive select statement shown below:

```
Select * from [GOSALES].SALES_TARGET
```

This statement is written in Cognos SQL, defines the scope of the query subject and generates run-time SQL when authoring a report or testing query subjects/items in Framework Manager.

5. Click the **Test** tab, and then in the bottom right corner, click **Test Sample**.

The data is retrieved and displayed in the Test results pane.

6. Click the **Query Information** tab.

This tab shows the SQL that was generated and used to retrieve the data that you saw in the Test results pane. The SQL is presented in both Cognos SQL and Native SQL as follows:

Cognos SQL	
select	
SALES_TARGET.SALES_YEAR as SALES_YEAR,	Projection list
SALES_TARGET.SALES_PERIOD as SALES_PERIOD,	
SALES_TARGET.COUNTRY_CODE_RETAILER as COUNTRY_CODE_RETAILER,	
SALES_TARGET.SALES_STAFF_CODE as SALES_STAFF_CODE,	
SALES_TARGET.RETAILER_CODE as RETAILER_CODE,	
SALES_TARGET.RETAILER_NAME as RETAILER_NAME,	
SALES_TARGET.PRODUCT_TYPE_CODE as PRODUCT_TYPE_CODE,	
SALES_TARGET.PRODUCT_BRAND_CODE as PRODUCT_BRAND_CODE,	
SALES_TARGET.SALES_TARGET as SALES_TARGET	
from	
GOSALES..GOSALES.SALES_TARGET SALES_TARGET	

Native SQL	
select "SALES_TARGET"."SALES_YEAR" "SALES_YEAR" ,	
"SALES_TARGET"."SALES_PERIOD" "SALES_PERIOD" ,	
"SALES_TARGET"."COUNTRY_CODE_RETAILER" "COUNTRY_CODE_RETAILER" ,	
"SALES_TARGET"."SALES_STAFF_CODE" "SALES_STAFF_CODE" ,	
"SALES_TARGET"."RETAILER_CODE" "RETAILER_CODE" ,	
"SALES_TARGET"."RETAILER_NAME" "RETAILER_NAME" ,	
"SALES_TARGET"."PRODUCT_TYPE_CODE" "PRODUCT_TYPE_CODE" ,	
"SALES_TARGET"."PRODUCT_BRAND_CODE" "PRODUCT_BRAND_CODE" ,	
"SALES_TARGET"."SALES_TARGET" "SALES_TARGET" from	
"GOSALES"."SALES_TARGET" "SALES_TARGET" FOR FETCH ONLY	

The syntax in the From clause of the Cognos SQL is composed of the following parts:

GOSALES..gosales.SALES_TARGET SALES_TARGET

Content Manager datasource Schema Database table IBM Cognos alias

The Cognos SQL is presented as an easy-to-read and formatted version of the native SQL. Viewing the native SQL gives a representation of what is actually sent to the database. The SQL in both versions select all columns individually rather than Select * from RETURNED_ITEM. This is because the SQL is generated based on the individual query items that make up the query subject. When you test the entire query subject, all query items are included in the query and therefore you see each column in the generated SQL.

Regardless of the type of SQL written on the SQL tab, Cognos SQL will always be displayed on the Query Information tab or be available in Report Studio for relational data sources.

7. Click **Cancel**.
8. Under **SALES_TARGET**, right-click the **SALES_TARGET** query item, click **Test**, and then click **Test Sample**.

9. Click the **Query Information** tab.

The results appear as follows:

<p>Cognos SQL</p> <pre>select SALES_TARGET.SALES_TARGET as SALES_TARGET from GOSALES..GOSALES.SALES_TARGET SALES_TARGET</pre>
<p>Native SQL</p> <pre>select "SALES_TARGET"."SALES_TARGET" "SALES_TARGET" from "GOSALES"."SALES_TARGET" "SALES_TARGET" FOR FETCH ONLY</pre>

Because you only selected one query item to test, only one column appears in the generated SQL. All unused items in the scope of the query subject have been dropped during optimization.

10. Click **Close**.

Task 2. Use a vendor specific function and configure the SQL Type setting.

1. In the **Project Viewer**, right-click the **gosales** namespace, point to **Create**, and then click **Query Subject**.
2. In the **Name** box, type **Current Year**, select **Data Source (Tables and Columns)**, and then click **OK**.
3. Under **Select a data source**, ensure **GOSALES** is selected, clear the **Run database query subject wizard** check box, and then click **Finish**.
4. Edit the SQL statement to appear as follows:

Select YEAR(current timestamp) "Current Year" FROM sysibm.sysdummy1

5. Click the **Test** tab.

An error message appears indicating there is a syntax error near "Year". Cognos SQL does not recognize this particular select statement. Cognos expects a From clause and a table name in the SQL statement. You will use the native SQL setting to leverage this statement.

6. Click **OK**, and then click **Options** in the lower right corner.
7. Click the **SQL Settings** tab, from the **SQL Type** list, select **Native**, click **OK** to the message, and then click **OK**.
8. Click **Test Sample**.

The current year based on the server date appears in the results pane.

You have successfully retrieved the current year using a vendor-specific function. You used native SQL due to the nature of your query. You did not select any columns from any of the tables in the database and again, Cognos SQL requires that you include "from" syntax and a table name to create a valid query subject. Native SQL lets you use bypass this rule. You simply asked the database to retrieve the current date and then modify it.

9. Click the **Query Information** tab.

The results appear as follows:

<p>Cognos SQL</p> <pre>select Current_Year."Current Year" as Current_Year from (GOSALES...{{select YEAR(current timestamp) "Current Year" from sysibm.sysdummy1}}) Current_Year</pre>
<p>Native SQL</p> <pre>select "Current_Year"."Current Year" "Current_Year" from (Select YEAR(current timestamp) "Current Year" from sysibm.sysdummy1) "Current_Year" FOR FETCH ONLY</pre>

The native SQL is reflected in the derived table portion of the Cognos SQL between the {} brackets. Derived tables will be discussed in further detail in another module.

10. Click **OK**, and then save the project.

Results:

By exploring the SQL tab and query information tab of a data source query subject, you saw the difference between the SQL that defines the scope of the query subject and the SQL that is generated at run time.

You also used a construct not permitted by Cognos SQL and changed the SQL Type setting to leverage the construct using a vendor specific function.

Benefits of Using Model Query Subjects

- Use model query subjects to:
 1. control query paths
 2. behave as views and control query generation
 3. simplify the model for presentation
 4. override settings specified for underlying query subjects
 5. resolve recursive relationships

1. You can control query paths by using model query subjects as aliases to prevent ambiguous joins.
2. You can control SQL generation with model query subjects by selecting query items from more than one underlying query subject and placing relationships on the model query subject. This creates As View behavior and will ensure underlying joins are honored in all query scenarios. This was seen earlier in the course when ORDER_HEADER and ORDER_DETAILS were merged together with relationships attached.

Once you implement relationships on a model query subject, you must implement all required relationships required for queries with that model query subject. You can not implement one relationship and also depend on underlying relationships to other query subjects to be used.

You can also control how a model query subject is viewed by the query engine (fact or dimension) based on the cardinalities you attach to it.

3. You can use model query subjects as a container for query items for a simplified business view. For example, you can combine product line, product type and product info into a simplified and more intuitive product dimension.
4. Model query subjects simply take a copy of the characteristics of the underlying objects on which they are based when they are created. They are independent of the underlying objects and can have their default behaviors changed without affecting the underlying objects. Changes to underlying objects will also not be reflected in the model query subjects.
5. You can also use model query subjects to resolve recursive relationships by creating an alias of a query subject and then relate it back to itself.

Override Relationships

- For different reporting requirements, you can use model query subjects to override relationship settings

Model

Query Subjects



Data Source

Query Subjects



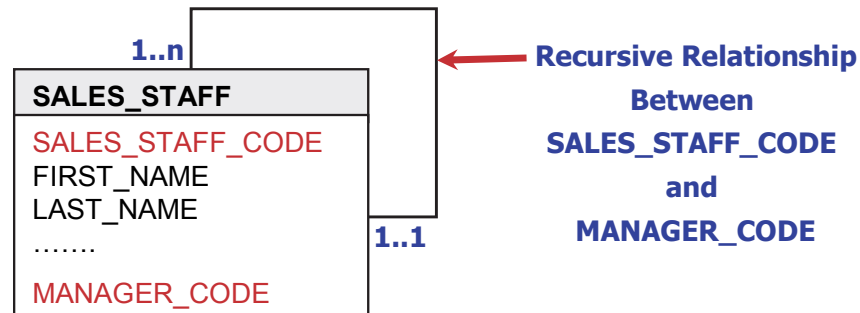
Using model query subjects, you can override underlying relationships to meet alternate reporting requirements. For example, in some reporting instances, authors would like to report only on employees who have sales targets. The data source query subjects' relationship meet this requirement because they have an inner join. But for other authors who would like to report on all employees regardless of whether they have sales targets or not, they can use the model query subjects to accomplish this since the cardinality has been changes to optional on the Sales Target Fact side.

Again, once you begin attaching relationships to model query subjects, you will need to create all required relationships for the query subject to meet your reporting needs. For example, if Sales Target Fact in the slide example will be queried with Time, then a relationship to Time would need to be created. The IBM Cognos query engine would not go back down to the data source query subject level to try and use the original relationship between SALES_TARGET_FACT and TIME_DIMENSION.

You can also use shortcuts to override underlying relationships.

Identify Recursive Relationships in Framework Manager

- Framework Manager displays self-joins in the diagram, but does not execute them as queries.

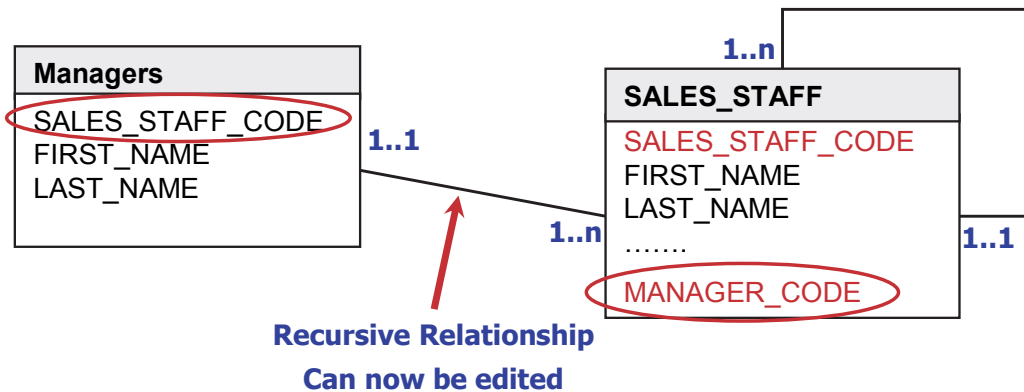


You must specify the self-join relationship at the data source level for the recursive relationship to be displayed in Framework Manager.

While you can view the metadata that defines the relationship, you cannot edit a recursive relationship in Framework Manager.

Resolve Recursive Relationships

- Use model query subjects (or shortcuts) to create and modify recursive relationships.



To modify a relationship that exists as a self-join in the data source, you can create a model query subject or shortcut and define a relationship between it and the original query subject.

Using the two query subjects in the slide example, you can create a master-detail query based on the same table in the data source.

Demo 2: Resolve a Recursive Relationship

Purpose:

Currently the GO Operational project uses the MANAGER query item from the EMPLOYEE_HISTORY query subject to report on an employee's manager. Authors would like to show the managers' contact information and report on managers and their employees.

To accomplish this, you will create an alias of the EMPLOYEE data source query subject and relate it to the EMPLOYEE_HISTORY data source query subject on MANAGER_CODE instead of EMPLOYEE code. This will allow you to use the EMPLOYEE table to retrieve manager names and their contact information.

Components: Framework Manager, Business Insight Advanced

Project: GO Operational

Package: GO Operational (query)

Task 1. Create an alias to resolve a recursive relationship in the data.

1. In Framework Manager, in the Project Viewer, under GO Operational Model>Foundation Objects View>gosales, create a model query subject called Manager.

2. Add the following query items:

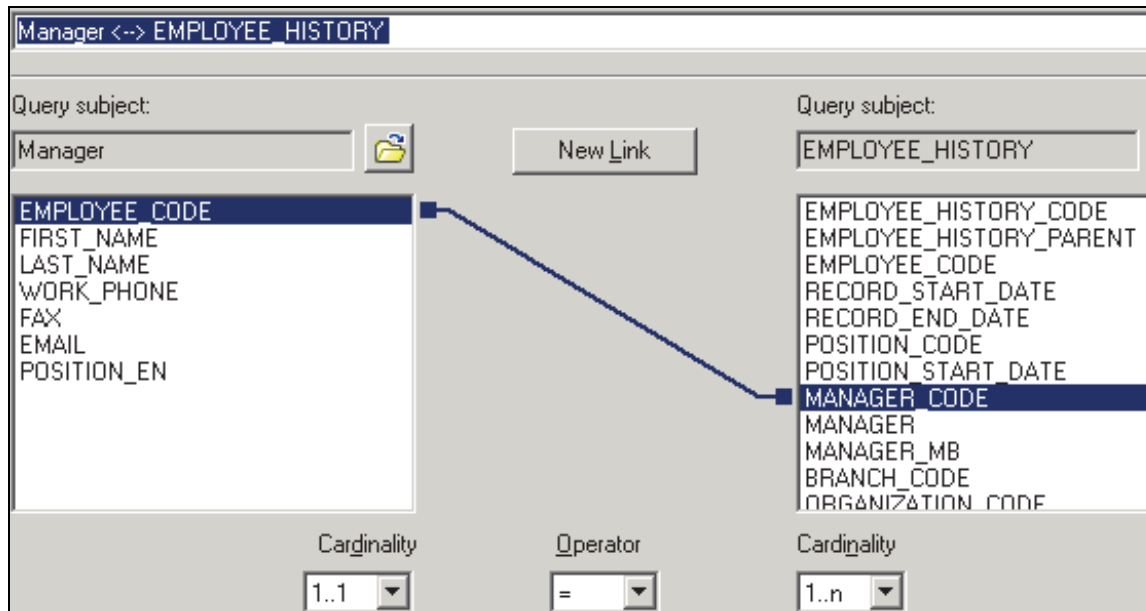
Query Subject	Query Item
EMPLOYEE	EMPLOYEE_CODE FIRST_NAME LAST_NAME WORK_PHONE EXTENSION FAX EMAIL
POSITION_LOOKUP	POSITION_EN

3. Click the ellipsis beside **POSITION_EN** and then change the expression definition as shown below to implement a language macro:

```
#'[gosales].[POSITION_LOOKUP].[POSITION_'
+$Language_lookup{$runLocale} + ']' #
```

4. Click **OK** twice, and then create a relationship from **Manager** (1..1, **EMPLOYEE_CODE**) to **EMPLOYEE_HISTORY** (1..n, **MANAGER_CODE**).

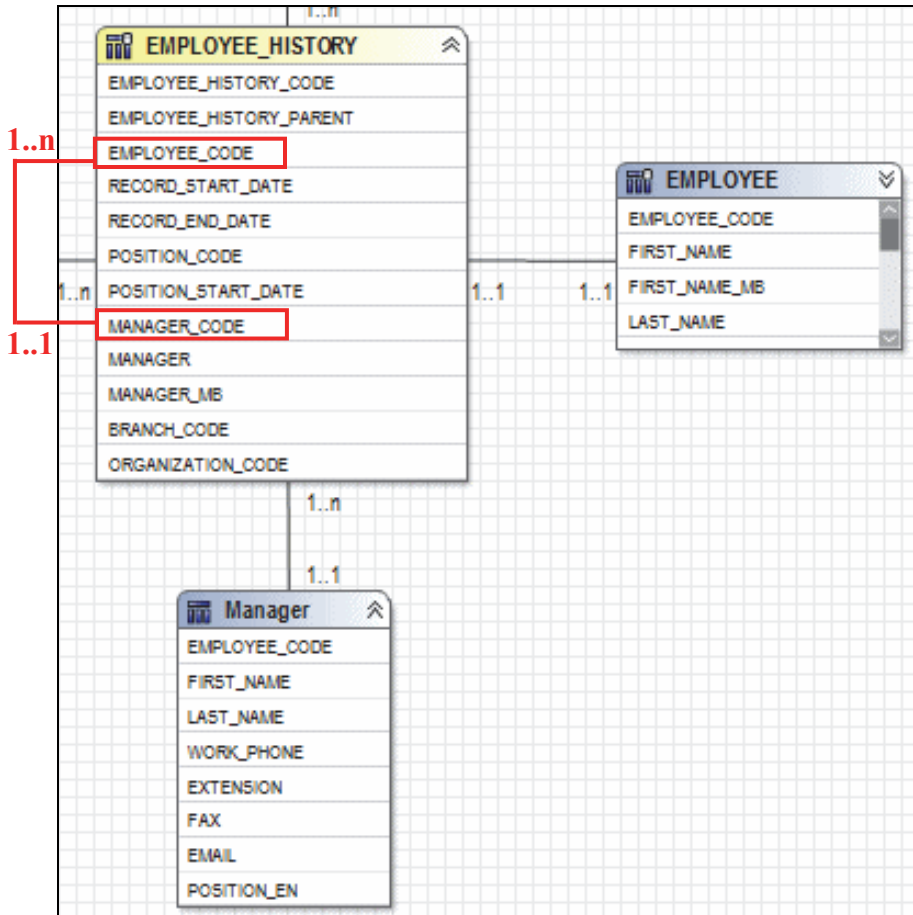
The results appear as follows:



5. Click **OK**, click **No** to the message, and then move the **Manager** model query subject below **EMPLOYEE_HISTORY** to group the items together.

- Right click **EMPLOYEE_HISTORY** and launch the **Context Explorer**, click the **Show related objects** button and then click the **Auto Layout** button to arrange in a star schema (you may need to adjust EMPLOYEE_HISTORY to see all query items).

The results appear as follows:

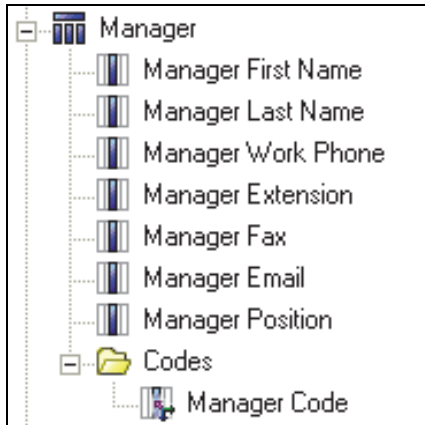


The Manager query subject can now query managers and the staff that report to them. It resolves the recursive nature of the data highlighted above.

Because POSITION_EN was placed in the Manager model query subject, that now has a relationship defined, As View behavior will occur. All underlying relationships will be honored in any query that uses the Manager query subject.

- Close the Context Explorer.

8. Rename and organize the query items in **Manager** as shown below:



Task 2. Incorporate query items from the Manager model query subject into the Consolidation View.

1. In the **Project Viewer**, expand **Consolidation View>Staff by Location**.
You will manually remap the Manager query item in the Staff by Location query subject to items in the new Manager model query subject in the Foundation Objects View. You want to display the full manager name so you will concatenate the manager's first and last name.
2. Double-click the **Manager** query item, and then delete the existing expression.
3. In the **Available Components** pane, expand **Foundation Objects View>gosales>Manager**.
4. Double-click **Manager First Name** to add it to the **Expression Definition** pane, and then type `|| ' ' ||` (there is a single space between the single quotes and no space between the pipes).
5. Double-click **Manager Last Name** to add it to the end of the expression.

The results appear as follows:

```
[gosales].[Manager].[Manager First Name] || ' ' || [gosales].[Manager].[Manager Last Name]
```

Note: You could also use + to concatenate the strings.

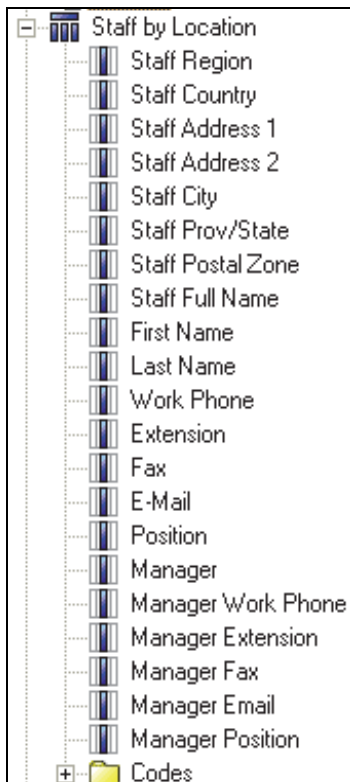
6. Click **Test Sample** .

The results appear similar to the following:

Test Results	
Manager	
Denis Pagé	
Élizabeth Michel	
Sherry Rowland	
Frank Bretton	
Georges Saint-Germain	
Carole Claudi	

7. Click **OK**, and then double-click **Staff by Location**.
8. In the **Available Model Objects** pane, expand **Foundation Objects View>gosales>Manager**.
9. Add the following items to the **Query Items and Calculations** pane:
- **Manager Work Phone**
 - **Manager Extension**
 - **Manager Fax**
 - **Manager Email**
 - **Manager Position**

10. Click **OK**, and then arrange the query items as shown below:



11. Save the project.

Task 3. Test the new Manager query subject in Business Insight Advanced.

1. Publish the **GO Operational (query)** package.
2. Launch **IBM Cognos Connection**, log on, and select **Author business reports** from the **Welcome** screen.
3. Select the **GO Operational (query)** package for a **List** report.

4. In the **Insertable Objects** pane, expand **Sales (query)**, and then add the following items to the report:

Query Subject	Query Item
Time	Year
Staff by Location	Manager Manager Position Staff Full Name Position
Sales Fact	Revenue

5. Group the report on **Year**, **Manager**, and **Manager Position**.

The results appear as follows:

Year	Manager	Manager Position	Staff Full Name	Position	Revenue
2004	Alex Rodriguez	Branch Sales Manager	Pierre Lavoie	Level 1 Sales Representative	\$3,795,932.90
			James Ripley	Level 3 Sales Representative	\$6,347,756.01
			Eric Carson	Level 3 Sales Representative	\$8,233,407.81
			Rhonda Cummings	Level 2 Sales Representative	\$10,224,806.94
		Branch Sales Manager - Summary			
	Alex Rodriguez - Summary				\$28,601,903.66
	Bayard Lopes	Branch Sales Manager	Beatriz Couto	Level 1 Sales Representative	\$626,268.72
			Alexandre Pereira	Level 3 Sales Representative	\$7,972,117.70
			Eduardo Guimarães	Level 3 Sales Representative	\$8,968,504.79
		Branch Sales Manager - Summary			
	Bayard Lopes - Summary				\$17,566,891.21
Cai Zhang	Branch Sales Manager	Xing Yu	Level 1 Sales Representative	\$4,944,971.88	
		Xiangyong Wang	Level 2 Sales Representative	\$6,900,577.09	

Managers are displayed along with staff that currently report to them and the revenue they have generated by year.

6. Return to **IBM Cognos Connection** without saving the report.

Results:

Using a model query subject to create an alias for the **EMPLOYEE** data source query subject you were able to resolve a recursive relationship found in **EMPLOYEE_HISTORY**.

Stored Procedure Query Subjects

- Two types:
 - Data Query:
 - return result sets based on simple to complex queries
 - must return a single uniform result set
 - Data Modification:
 - modify the database
 - used in Event Studio only
- Both types may expect arguments

For Data Query stored procedure query subjects, Framework Manager can leverage the stored procedure by generating a query subject that reflects the returned result set.

If a stored procedure returns multiple result sets, IBM Cognos only supports the first result set. IBM Cognos will define the metadata according to the result set returned by the stored procedure when it is first created in Framework Manager. An error is generated at run time if the stored procedure returns a different result set than originally defined when the stored procedure query subject was created.

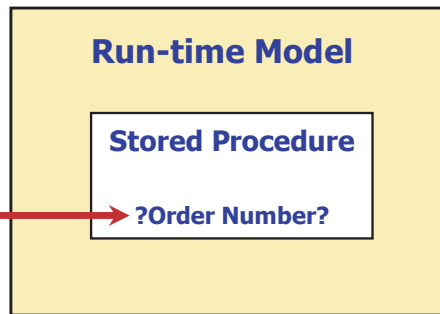
Some data source systems allow for multiple stored procedures with the same name, but each accepts a different number and/or type of arguments. The number and type of arguments passed determine which stored procedure is used. This is known as an overloaded signature. To work with overloaded signatures, create multiple stored procedures, each with a unique name, and create a separate query subject for each result set or signature.

Output parameters are not supported.

Data Modification stored procedure query subjects let you leverage a stored procedure from the data source to update data in the data source. These types of stored procedure query subjects are only available for use in Event Studio.

Use Prompt Values

- A prompt value can be used when user input is required for variables beyond the report author's control.
- The syntax for using a prompt as a value is:
 - ?Prompt Name?



In general, it is better to define prompts in the reporting application to make use of the additional prompt features. However, there are some variables that report authors cannot modify such as parameters in a stored procedure. For these, you can use Framework Manager to define prompts.

Prompt values can also be used in:

- parameter maps
- session parameters
- expressions including filters, calculations, and relationships

Demo 3: Create and Test a Data Query Stored Procedure Query Subject

Purpose:

Management has requested a quick reference tool for the IBM Cognos reporting environment to allow phone representatives at the call center to quickly retrieve information for specific orders. A stored procedure exists in the GOSALES database that can be leveraged to create this tool.

To accomplish this, you will create a stored procedure query subject and configure it appropriately.

Components: Framework Manager, Business Insight Advanced

Project: GO Operational

Package: GO Call Center

Task 1. Create a stored procedure query to retrieve data.

1. In the **Project Viewer**, create a folder called **Stored Procedures** in the **gosales** namespace.
2. Right-click the **Stored Procedures** folder, point to **Create**, and then click **Query Subject**.
3. In the **Name** box, type **Find Order Information**, select **Stored Procedure**, and then click **OK**.
4. In the **Select a data source** pane, ensure **GOSALES** is selected, and then click **Next**.
5. In the **Stored Procedures** pane, expand **GOSALES>GOSALES>Procedures**.

6. Click the **FINDORDERINFO** stored procedure, and then click **Finish**.

The Query Subject Definition window for the stored procedure appears.

Note: You can also import a stored procedure using the Metadata Import Wizard. If you do, initially the query subject will appear broken. You must edit the query subject to verify its projection list. After testing the query subject and retrieving the projection list, the query subject appears normal in the Project Viewer.

You will now add a prompt value to allow for user input as opposed to hard coding a value for the **ORDERNUMBER** argument.

7. Click the **ORDERNUMBER** argument, and then click the **ellipsis** in the **Value** column.
8. In the **Value** pane, type **?Order Number?**.
9. Click **OK**, and then click the **Test** tab.

The Prompt Values dialog box appears. In order to prevent continually being prompted, you will clear the Always prompt for values when testing box.

10. Click in the **Value** field, type **100002**, clear the **Always prompt for values when testing**, and then click **OK**.
11. Click **Test Sample** in the bottom right corner.

The results appear as follows:

Test Results					
	ORDER_NUMBEF	RETAILER_NAME	PRODUCT_NUMBER	ORDER_DATE	SHIP_DATE
	100002	Ar fresco	75110	Jan 12, 2004 12:00:00 AM	Jan 19, 2004 12:00:00 AM
	100002	Ar fresco	76110	Jan 12, 2004 12:00:00 AM	Jan 19, 2004 12:00:00 AM
	100002	Ar fresco	85110	Jan 12, 2004 12:00:00 AM	Jan 19, 2004 12:00:00 AM
	100002	Ar fresco	65110	Jan 12, 2004 12:00:00 AM	Jan 19, 2004 12:00:00 AM
	100002	Ar fresco	100110	Jan 12, 2004 12:00:00 AM	Jan 19, 2004 12:00:00 AM

Several records are returned in the Test Results pane with related order information.

12. Click **OK**, and then save the project.

Task 2. Edit query item properties and create a relationship PRODUCT_NAME_LOOKUP.

The stored procedure returns a product number but not a product name, which would be more useful for your application. You can create a relationship to the **PRODUCT_NAME_LOOKUP** query subject using **PRODUCT_NUMBER** to later obtain the appropriate product name.

1. Under the **Stored Procedures** folder, expand **Find Order Information**.
2. Ctrl+click **ORDER_NUMBER**, and **PRODUCT_NUMBER**, and then, in the **Properties** pane, change the **Usage** property for both query items to **Identifier**.

In this case, you know that the underlying fields in the database are in fact indexed and can therefore be set as identifiers instead of attributes.

3. In the **Project Viewer** under **gosales**, click **PRODUCT_NAME_LOOKUP**, and then Ctrl+click **Find Order Information**.
4. Right-click either query subject, and then click **Create Relationship**.

5. Change the relationship to be from **PRODUCT_NAME_LOOKUP 1.1** to **Find Order Information 1.1** on **PRODUCT_NUMBER**, as shown below:

The screenshot shows the IBM Query Editor interface. At the top, the 'Name' field contains 'PRODUCT_NAME_LOOKUP <--> Find Order Information'. Below this, there are two 'Query subject' sections. The left section is for 'PRODUCT_NAME_LOOKUP' and the right section is for 'Find Order Information'. A blue line connects the 'PRODUCT_NUMBER' field in the left list to the 'PRODUCT_NUMBER' field in the right list. Below the lists, there are three dropdown menus: 'Cardinality' (set to '1..1'), 'Operator' (set to '='), and 'Cardinality' (set to '1..1').

PRODUCT_NAME_LOOKUP, in this case, will act as a lookup table for Find Order Information to retrieve product name values.

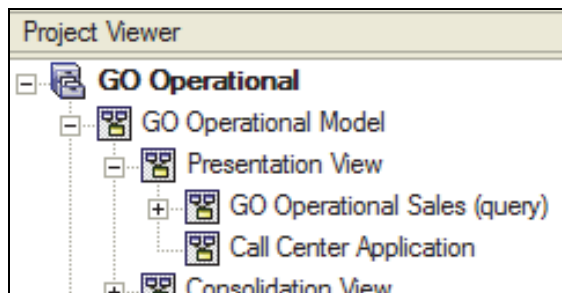
6. Click **OK**, and then save the project.

Task 3. Create a Find Order Information model query subject.

Now that you have a relationship between Find Order Information and PRODUCT_NAME_LOOKUP, you can create a model query subject that retrieves all the required information for your call center application. You will create a business view for this application.

1. In the **Project Viewer**, in the **Presentation View** namespace, create a new namespace called **Call Center Application**.

The results appear as follows:



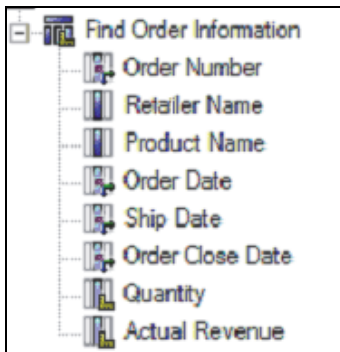
2. In the **Call Center Application** namespace, create a new model query subject called **Find Order Information**, and then click **OK**.
3. In the **Available Model Objects** pane, expand **Foundation Objects View>gosales>Stored Procedures>Find Order Information**.
4. Add all query items to the **Query Items and Calculations** pane except **PRODUCT_NUMBER**.
5. Expand **PRODUCT_NAME_LOOKUP**, and then add **PRODUCT_NAME** to the **Query Items and Calculations** pane.
6. Arrange **PRODUCT_NAME** so that it is the third item in the list using the green arrows on the right side of the dialog box.

7. Click the **Test** tab, and then click **Test Sample** in the bottom right corner.

The order information appears with the appropriate product name, as shown below:

Test results				
	ORDER_NUMBEF	RETAILER_NAME	PRODUCT_NAME	ORDER_DATE
	100002	Ar fresco	Mountain Man Deluxe	Jan 12, 2004 12:00:00 AM
	100002	Ar fresco	Edge Extreme	Jan 12, 2004 12:00:00 AM
	100002	Ar fresco	Bear Edge	Jan 12, 2004 12:00:00 AM
	100002	Ar fresco	Glacier GPS Extreme	Jan 12, 2004 12:00:00 AM
	100002	Ar fresco	Insect Bite Relief	Jan 12, 2004 12:00:00 AM

8. Click **OK**.
9. Expand **Find Order Information**, and then, if time permits rename the query items as shown below:

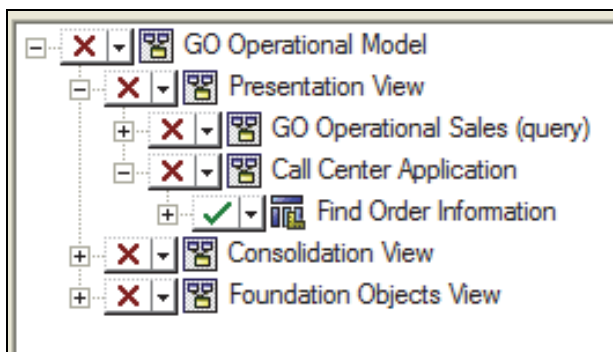


If time is short, you can skip renaming all query items as this will be done for you at the start of the next module.

Task 4. Create a new package for the call center phone representatives.

1. In the **Project Viewer**, create a package called **GO Call Center** that only contains **Find Order Information** from the **Call Center Application** namespace.

The results appear as follows:



2. Click **Finish**.

You are prompted to open the Publish Package wizard.

3. Click **Yes**.
4. Clear the **Enable model versioning** check box, click **Next**, click **Next**, click **Publish**, and then click **Finish**.

The Verify Model dialog appears listing informational messages.

5. Click **Close**, and then save the project.

Task 5. Test the Find Order Information query subject in Business Insight Advanced.

1. Launch **IBM Cognos Connection**, log in, and then launch **Business Insight Advanced** selecting the **GO Call Center** package for a **List** report.
2. Drag the **Find Order Information** query subject to the report.

An Order Number prompt appears.

3. In the **Provide a number** box, type **100004**, and then click **OK**.

A list appears displaying all the records for the requested order number, as shown below:

Order Number	Retailer Name	Product Name	Order Date	Ship Date	Order Close date	Quantity	Actual Revenue
100004	Ao ar livre	Hibernator Lite	Jan 12, 2004 12:00:00 AM	Jan 21, 2004 12:00:00 AM	Jan 21, 2004 12:00:00 AM	354	29,658.12
100004	Ao ar livre	Star Gazer 2	Jan 12, 2004 12:00:00 AM	Jan 21, 2004 12:00:00 AM	Jan 21, 2004 12:00:00 AM	139	75,289.35
100004	Ao ar livre	Star Lite	Jan 12, 2004 12:00:00 AM	Jan 21, 2004 12:00:00 AM	Jan 21, 2004 12:00:00 AM	261	89,841.42
100004	Ao ar livre	TrailChef Deluxe Cook Set	Jan 12, 2004 12:00:00 AM	Jan 21, 2004 12:00:00 AM	Jan 21, 2004 12:00:00 AM	279	33,658.56
Overall - Summary						1,033	228,447.45

4. Return to **IBM Cognos Connection**, do not save changes, and leave Framework Manager open for the next demo.

Results:

By incorporating a stored procedure query subject with a prompt value into the model you were able to provide an easy-to-use call center application.

Demo 4: Create and Test a Data Modification Stored Procedure Query Subject

Purpose:

A request has come in to make a stored procedure available from the GOSALES database for use in Event Studio. This stored procedure updates columns in the RETURNED_ITEM table. Authors would like to automate a process to notify sales representatives when an order has a return. The automated process will then assign the sales representative responsible for the order by updating the RETURNED_ITEM table. The sales representative is then expected to follow up with the customer.

To accomplish this, you will create a stored procedure query subject and configure it appropriately.

Component: **IBM DB2 Control Center, Framework Manager**

Project: **GO Operational**

Stored procedures can update databases such as adding, updating, or deleting records.

Business users can use Event Studio to author agents that monitor changes in your data based on defined conditions. When conditions are met, the agent can run a series of tasks, including an update database task. This task is defined by leveraging a stored procedure that you can include in a model and publish as part of a package.

There are typically limited scenarios in which this technique is used. Most organizations do not allow external applications to update source data. In any application that does use a stored procedure to update data, the database administrator has control over supplying a procedure that limits the update to specified columns and tables in the data source.

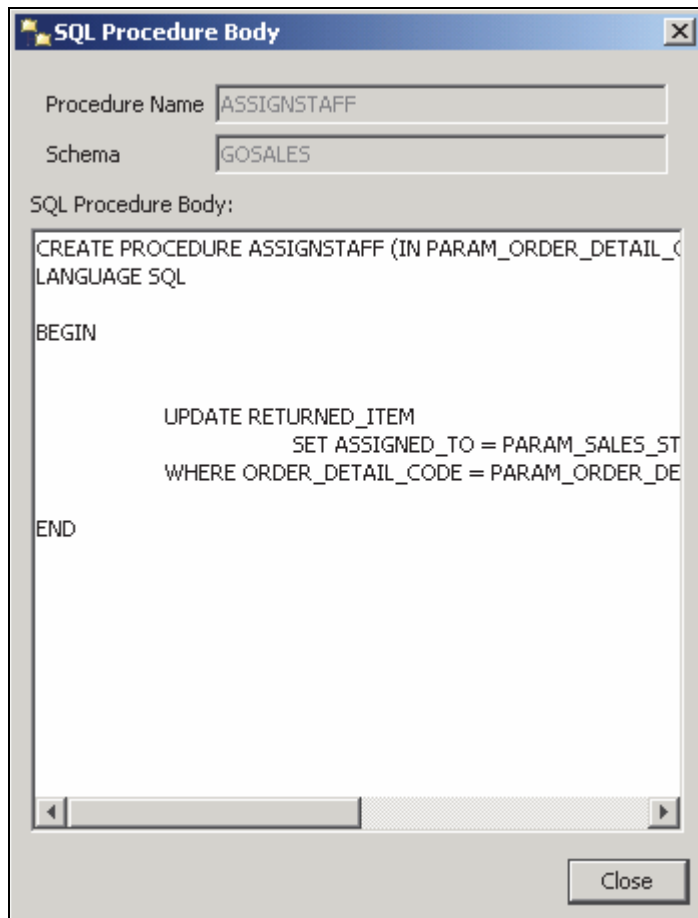
IBM Cognos places restrictions on the use of stored procedures that update data in a data source. They can only be leveraged from within an Event Studio agent. They cannot be accessed in any other studio.

Task 1. Examine the AssignStaff stored procedure in IBM DB2.

1. From the **Start** menu, point to **All Programs>IBM DB2>DB2COPY1 (Default)>General Administration Tools** and then click **Control Center**.
2. Select the **Advanced** radio button and click **OK**.
3. Expand **All Databases>GS_DB>Application Objects**, and then click **Stored Procedures**.
4. Scroll to the **ASSIGNSTAFF** stored procedure visible in the right pane.

5. Right-click the **ASSIGNSTAFF** stored procedure, and then click **Show SQL Procedure Body**.

The Show SQL dialog box appears as shown below:



This stored procedure updates the RETURNED_ITEM table. It adds the sales staff code to the ASSIGNED_TO column, and retrieves and adds the system date to the DATE_ADVISED column. It also has two arguments that must be supplied: SALES_STAFF_CODE and ORDER_DETAIL_CODE.

6. Click **Close**, and then close **Control Center**.

Task 2. Test the RETURNED_ITEM query subject.

1. In **Framework Manager**, in the **Project Viewer**, if necessary, expand the **Original Sales & Returns Objects** folder.
2. Test the **RETURNED_ITEM** query subject.
3. Scroll to the right until you see the **ASSIGNED_TO** column.

The results appear as follows:

Test results					
	ORDER_DETAIL_CODE	RETURN_REASON_CODE	RETURN_QUANTITY	ASSIGNED_TO	FOLLOW_UP_CODE
	4000059	1	30		-1
	6000325	4	152		-1
	5000097	1	146		-1
	80214477	4	47		-1

Notice that the **ASSIGNED_TO** values are null. You will import a stored procedure to update columns in this table and test for the **ORDER_DETAIL_CODE** value of 4000059, the first row in this table.

4. Click **Close**.

Task 3. Import the stored procedure into the model.

1. Right-click the **Stored Procedures** folder, point to **Create**, and then click **Query Subject**.
2. In the **Name** box, type **AssignStaff**, select **Stored Procedure**, and then click **OK**.
3. In the **Select a data source** pane, ensure **GOSALES** is selected, and then click **Next**.
4. In the **Stored Procedures** pane, expand **GOSALES>GOSALES>Procedures**.

5. Click the **ASSIGNSTAFF** stored procedure, and then click **Finish**.

The Query Subject Definition window for the stored procedure appears.

You will now add prompt values to allow for user input as opposed to hard coding values for the expected arguments.

6. Click the **PARAM_SALES_STAFF_CODE** argument, and then click the **ellipsis** in the **Value** column.
7. In the **Value** pane, type **?SalesStaffCode?**.
8. Click **OK**, and then add the following prompt value for the **PARAM_ORDER_DETAIL_CODE** argument:

?OrderDetailCode?

The results appear as follows:

Argument Name	Mode	Type	Format	Value
PARAM_ORDER_DETAIL_CODE	in	characterLength16	Size=34, Precision=16, Scal	?OrderDetailCode?
PARAM_SALES_STAFF_CODE	in	int32	Size=4, Precision=0, Scale=	?SalesStaffCode?

9. Click the **Test** tab.

The Prompt Values dialog box appears.

10. Click in the **Value** field for **OrderDetailCode**, type **4000059**, click the **Value** field for **SalesStaffCode** twice, type **572**, and then click **OK**.

An error message appears stating the stored procedure is unable to return a result set. This is because the stored procedure updates a table and does not retrieve rows from a table. You will modify the definition of this stored procedure query subject to act as a data modification stored procedure query subject.

11. Click **OK**, click **Cancel**, and then click the **Definition** tab.

In the Type box, notice that the current setting is Data Query. With this setting, the stored procedure, when tested, should return a result set.

12. Change the **Type** setting to **Data Modification**.
13. Click the **Test** tab, and then click **Test Sample**.
14. Click **OK**.

A message appears stating the stored procedure executed successfully.

15. Click **OK** three times, and then save the project.

Task 4 Re-test the RETURNED_ITEM query subject.

1. In the **Original Sales & Returns Objects** folder, test the **RETURNED_ITEM** query subject.
2. Scroll to the right until you see the **ASSIGNED_TO** column.

The results appear as follows:

Test results				
ORDER_DETAIL_CODE	RETURN_REASON_CODE	RETURN_QUANTITY	ASSIGNED_TO	FOLLOW_UP_CODE
4000059	1	30	572	-1
6000325	4	152		-1
5000097	1	146		1

Notice that the ASSIGNED_TO value in the first row is now 572, the value you provided in the prompt dialog. If you scroll right a little further, you will see that the DATE ADVISED column has a value for the current date and time. The row for ORDER_DETAIL_CODE 4000059 was successfully updated.

3. Click **Close**, and leave Framework Manager open for the next demo.

This stored procedure can now be added to a package and used in Event Studio.

Results:

By creating a stored procedure query subject and configuring it appropriately, you successfully created and tested a query subject that can be leveraged in Event Studio to update the GOSALES database.

Modify Relationships

- You can modify the:

- link(s)
- cardinality
- operator
- expression

Cognos.
software

© 2010 IBM Corporation

You can choose which key or keys link two query subjects together, control their cardinality, and specify different operators.

You can also create complex join conditions in the Expression editor. For example, you may want to extend the join syntax to include other filter criteria. This way the filter is applied only if items from both query subjects are used, and leaves the individual query subjects unrestricted in other query scenarios.

Modifying any of the relationship settings is reflected in the generated SQL. For example, optional cardinality on one end will generate a left outer join in the generated SQL, and optional cardinality on both ends will generate a full outer join in the generated SQL.

Demo 5: Create a Compound Relationship Expression

Purpose:

Currently there is a filter on the **EMPLOYEE_HISTORY** data source query subject. This filter limits the records to each employee's current record. Because a filter has been applied to this query subject, its record set is restricted in all query scenarios. Human Resources have requested the ability to report on employee history.

To accomplish this, you will move the filter from the **EMPLOYEE_HISTORY** data source query subject to the relationship expression between **EMPLOYEE_HISTORY** and **EMPLOYEE**. You will then create a model query subject as an alias to **EMPLOYEE** with a relationship to **EMPLOYEE_HISTORY** so that human resources staff can author their reports.

Component: Framework Manager

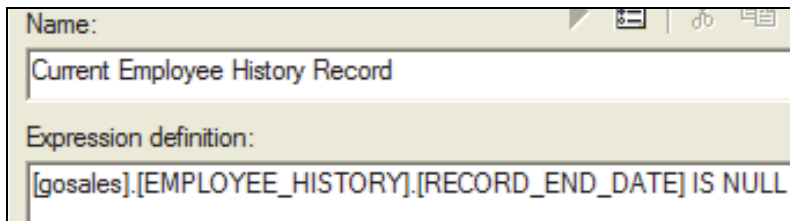
Project: GO Operational

Task 1. View the existing **EMPLOYEE_HISTORY filter and move it to a relationship expression.**

1. In the **Project Viewer**, double-click **EMPLOYEE_HISTORY**, and then click the **Filters** tab.

2. In the **Source** column beside the **Current Employee History Record** filter, click the **ellipsis**.

The results appear as follows:



Name: Current Employee History Record

Expression definition: [gosales].[EMPLOYEE_HISTORY].[RECORD_END_DATE] IS NULL

This is a filter applied on a data source query subject to retrieve only the most recent record per employee (the one with no End Date assigned).

3. Copy the expression in the **Expression definition** pane, and then click **Cancel**.
4. Delete the filter, and then click **OK**.

You will now test the effect of removing the filter.

5. Select and test the following items:

Query Subject	Query Item
EMPLOYEE	EMPLOYEE_CODE FIRST_NAME LAST_NAME
EMPLOYEE_HISTORY	EMPLOYEE_HISTORY_CODE MANAGER

Some employees have more than one record. These are their current and historical records. This is the type of reporting human resources staff would like to do.

6. Click **Close**, right-click **EMPLOYEE_HISTORY**, click **Launch Context Explorer**, and then click **Show Related Objects**.
7. Double-click the relationship between **EMPLOYEE_HISTORY** and **EMPLOYEE**.
8. Click the **ellipsis** beside the **Expression** pane, at the end of the expression, type **and**, and then paste the syntax from the filter after the word **and**.
9. Click **OK**.

The results appear as follows:

Name: EMPLOYEE <--> EMPLOYEE_HISTORY

Query subject: EMPLOYEE

Query subject: EMPLOYEE_HISTORY

EMPLOYEE_CODE
FIRST_NAME
FIRST_NAME_MB
LAST_NAME
LAST_NAME_MB
DATE_HIRED
TERMINATION_DATE
TERMINATION_CODE
BIRTH_DATE
GENDER_CODE
WORK_PHONE
EXTENSION

EMPLOYEE_HISTORY_CODE
EMPLOYEE_HISTORY_PARENT
EMPLOYEE_CODE
RECORD_START_DATE
RECORD_END_DATE
POSITION_CODE
POSITION_START_DATE
MANAGER_CODE
MANAGER
MANAGER_MB
BRANCH_CODE
ORGANIZATION_CODE

Cardinality: 1..1 Operator: = Cardinality: 1..1

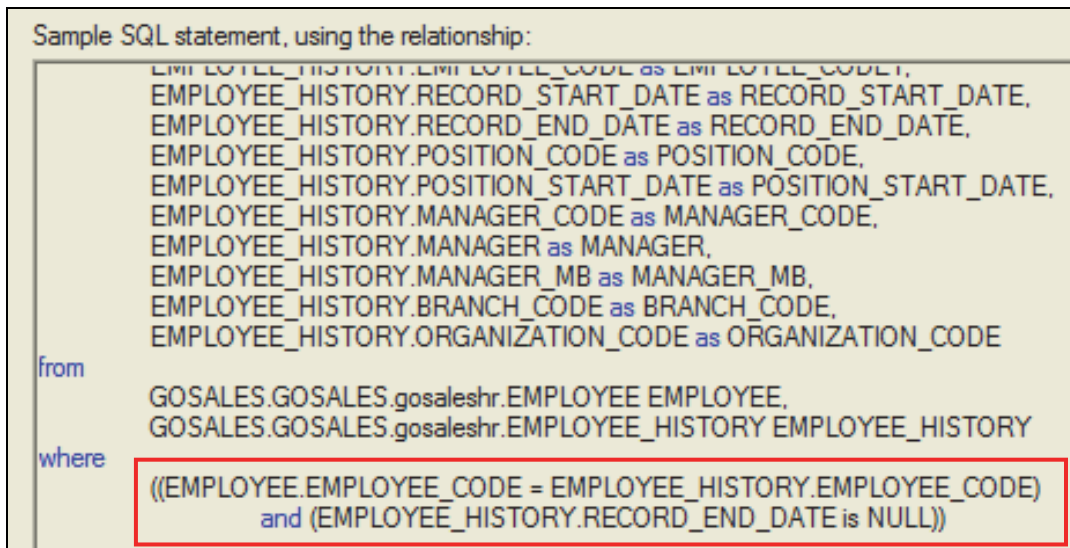
Relationship impact: Each EMPLOYEE_HISTORY has one and only one EMPLOYEE.
Each EMPLOYEE has one and only one EMPLOYEE_HISTORY.

Expression:
EMPLOYEE.EMPLOYEE_CODE = EMPLOYEE_HISTORY.EMPLOYEE_CODE and
EMPLOYEE_HISTORY.RECORD_END_DATE IS NULL

The link between the two query subjects is no longer present since the expression contains non-join specific syntax.

10. Click the **Relationship SQL** tab, and then scroll down until you see the **where** clause.

The results appear as follows:



Sample SQL statement, using the relationship:

```

EMPLOYEE_HISTORY.EMPLOYEE_CODE as EMPLOYEE_CODE,
EMPLOYEE_HISTORY.RECORD_START_DATE as RECORD_START_DATE,
EMPLOYEE_HISTORY.RECORD_END_DATE as RECORD_END_DATE,
EMPLOYEE_HISTORY.POSITION_CODE as POSITION_CODE,
EMPLOYEE_HISTORY.POSITION_START_DATE as POSITION_START_DATE,
EMPLOYEE_HISTORY.MANAGER_CODE as MANAGER_CODE,
EMPLOYEE_HISTORY.MANAGER as MANAGER,
EMPLOYEE_HISTORY.MANAGER_MB as MANAGER_MB,
EMPLOYEE_HISTORY.BRANCH_CODE as BRANCH_CODE,
EMPLOYEE_HISTORY.ORGANIZATION_CODE as ORGANIZATION_CODE
from
GSALES.GSALES.gosaleshr.EMPLOYEE EMPLOYEE,
GSALES.GSALES.gosaleshr.EMPLOYEE_HISTORY EMPLOYEE_HISTORY
where
((EMPLOYEE.EMPLOYEE_CODE = EMPLOYEE_HISTORY.EMPLOYEE_CODE)
and (EMPLOYEE_HISTORY.RECORD_END_DATE is NULL))

```

Your compound expression is reflected in the generated SQL.

11. Click the **Test** button.

Multiple records per employee no longer exist. Records for the employee codes 10016 and 10020 only appear once. This filter will only be implemented when items from both query subjects are used in a query or the relationship between them is used in a query path. For example, if your query contained an item from EMPLOYEE_HISTORY and SALES_TARGET_FACT, EMPLOYEE will be in the query path and therefore the filter will be applied.

12. Click **OK**, and then close the **Context Explorer**.

Task 2. Create an alias for EMPLOYEE using a model query subject.

Human resources staff will use this new alias to report on historical employee data without the restriction of the filter that you removed from the EMPLOYEE_HISTORY data source query subject.

1. In the **Project Viewer**, right-click **EMPLOYEE**, and then click **Merge in New Query Subject**.
2. Click **No** to creating the underlying relationships.
3. Rename the new **EMPLOYEE_EMPLOYEE** model query subject to **Employee (Human Resources) (alias)**.
4. Create a relationship between **Employee (Human Resources) (alias) (1..1)** and **EMPLOYEE_HISTORY (1..n)** on **EMPLOYEE_CODE**.
5. Click **OK**, and then click **No** when asked to create other underlying relationships.

6. Select and test the following items:

Query Subject	Query Item
Employee (Human Resources) (alias)	EMPLOYEE_CODE FIRST_NAME LAST_NAME
EMPLOYEE_HISTORY	EMPLOYEE_HISTORY_CODE MANAGER

The results appear as follows:

Test results				
EMPLOYEE_CODE	FIRST_NAME	LAST_NAME	EMPLOYEE_HISTORY_CODE	MANAGER
10359	Aaltje	Hansen	30336	Ulla Blackmore
10004	Denis	Pagé	30000	Dietz Krieger
10005	Élizabeth	Michel	30001	Frédéric Samson
10006	Émile	Clermont	30002	Frédéric Samson
10007	Étienne	Jauvin	30003	Frédéric Samson
10012	Elsbeth	Wiesinger	30004	Else Mörike
10013	Else	Mörike	30005	Barbara Samuelsen
10014	Frank	Fuhlroth	30006	Georges Saint-Germain
10015	Gunter	Eder	30007	Elsbeth Wiesinger
10016	Björn	Winkler	30956	Gretchen Goetschy
10016	Björn	Winkler	30008	Fritz Hirsch
10017	Fritz	Hirsch	30009	Else Mörike
10018	Jörg	Kunze	30010	Denis Pagé
10019	Silvano	Allessori	30011	Maria Iacobucci
10020	Maria	Iacobucci	30938	Derek Hirsch
10020	Maria	Iacobucci	30012	Fabian Tibor
10021	Alessandra	Torta	30013	Maria Iacobucci
10022	Belinda	Jansen-Velasquez	30014	Kick Kalkman
10023	Ellen	Shapiro	30015	Kick Kalkman

The historical records are now returned when querying between the employee alias query subject and EMPLOYEE_HISTORY and can be leveraged by the human resources staff.

7. Click **Close**, and then save and close the project.
8. If necessary, close any open browser windows and then close Framework Manager.

Results:

By moving a filter out of the **EMPLOYEE_HISTORY** data source query subject and into the relationship expression between **EMPLOYEE_HISTORY** and **EMPLOYEE**, you extended the model's reporting capabilities where employees are concerned. You created an alias for **EMPLOYEE** with a model query subject, and changed its relationship configuration to **EMPLOYEE_HISTORY** so that human resources staff can report on all employee data.

Considerations When Modifying Query Subjects

- When modifying any query subject or relationship, view the generated SQL to verify your intended modeling technique.
- Modifying data source query subjects may make them "complex" and cause IBM Cognos to request metadata from the database.
- When you add relationships to model query subjects, encapsulated joins are honored.

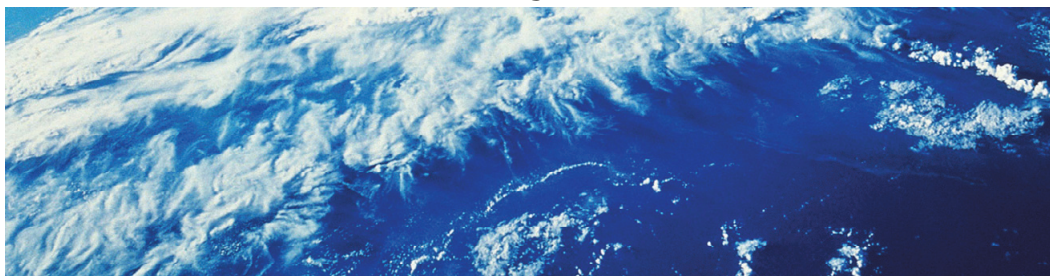
Summary

- You should now be able to:
 - identify key differences and make recommendations for using data source, model, and stored procedure query subjects
 - identify the effects on generated SQL when modifying query subjects, SQL settings and relationships



Set Security in Framework Manager

IBM Cognos BI

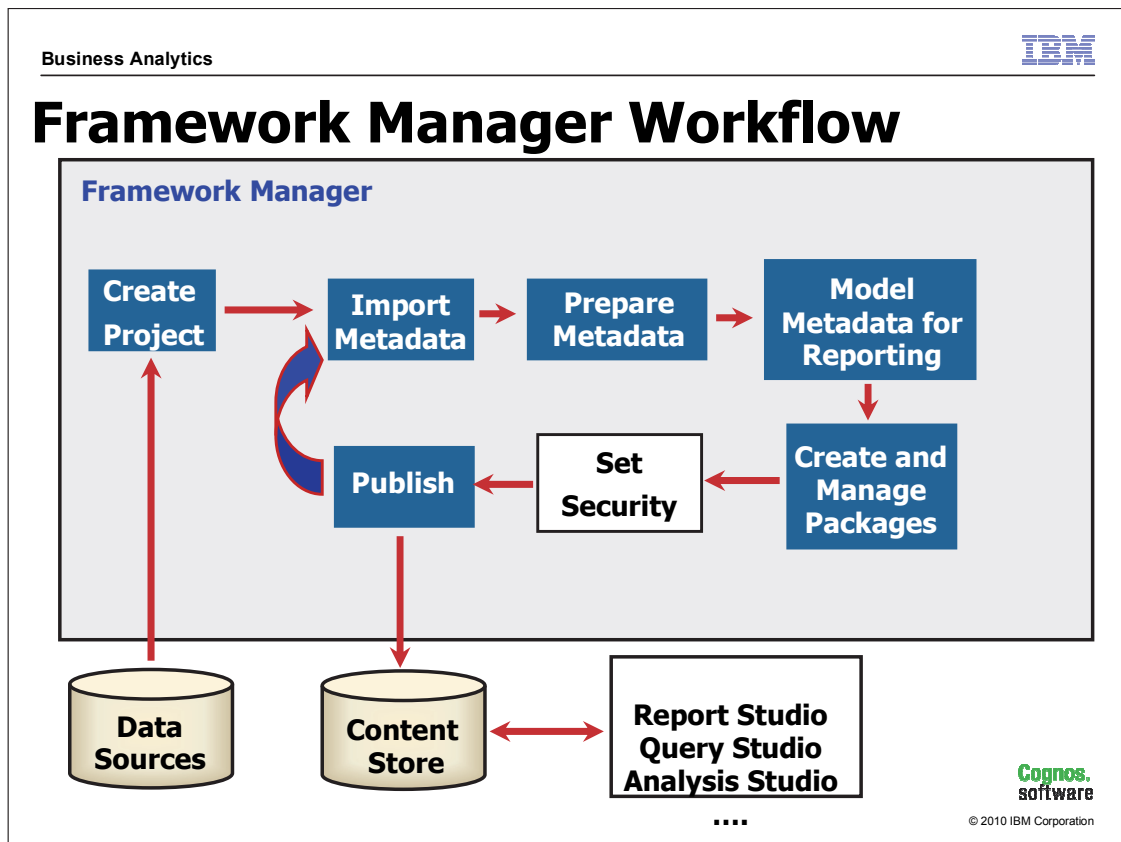


Business Analytics

© 2010 IBM Corporation

Objectives

- At the end of this module, you should be able to:
 - examine the IBM Cognos security environment
 - restrict access to packages
 - create and apply security filters
 - restrict access to objects in the model



This module deals with applying security to your model in Framework Manager based on security requirements defined for your IBM Cognos environment.

Examine IBM Cognos Security

- Consists of:
 - third-party authentication providers
 - used to authenticate users and secure objects
 - Cognos namespace
 - used to create IBM Cognos specific groups and roles to secure objects
 - capabilities and user interface profiles
 - used to grant or deny users access to IBM Cognos components or features
- Security is optional - anonymous access is allowed

IBM Cognos can leverage one or more third-party authentication providers to authenticate users. You can use the providers to define and maintain users, groups, and roles as required.

In addition IBM Cognos provides its own namespace called Cognos. The Cognos namespace contains groups and roles that, by default, define user privileges in the IBM Cognos environment. You can use this namespace to enhance your organization's security policies and ensure the portability of your applications. For example, you can exclusively use Cognos namespace groups and roles to secure your application and simply add users or groups from a third-party authentication provider. If you port your application to a different environment, you can continue to use the Cognos groups and roles and add the appropriate users or groups from the new authentication provider.

Set Security in Framework Manager

- When you apply security in Framework Manager, you control access for selected users, groups, and roles.
- Grant or deny access for:
 - packages
 - data
 - objects

Package access refers to the ability to use the package in an IBM Cognos studio (for example, Query Studio), or to run a report that uses the package from IBM Cognos Connection. Users without these abilities are denied access, although they can still view saved report outputs if they have access to the reports. You can also give administrative access to packages for users who are required to republish packages or perform impact analysis based on model changes.

Data security restricts the data returned by query subjects. You create a security filter that is placed on a specific query subject and applied to specific users, groups, or roles, that will be using that query subject.

When you add object security, you are restricting access to objects, such as query subjects, query items, and filters.

Permissions

- When you specify permissions for objects, you allow or deny the following permissions:
 - Read
 - Write
 - Execute
 - Set Policy
 - Traverse

Read: View all the properties of an entry.

Write: Modify properties of an entry. Delete an entry. Create entries in a container, such as a package or a folder. Modify the report specification for reports created in Report Studio and Query Studio. Create new outputs for a report.

Execute: Process an entry such as running a report or retrieving data through data source connections.

Set Policy: Read and modify the security settings for an entry.

Traverse: View the contents of a container entry, such as a package or a folder, and view general properties of the container itself without full access to the content.

Demo 1: Specify Package Access

Purpose:

You want to ensure that only Query Users and Authors can access the GO Operational (query) package and that only System Administrators can administer the package.

You will view the results in IBM Cognos Connection by logging in as a systadmem administrator and then as a query user and author.

Components: Framework Manager, IBM Cognos Connection

Project: GO Operational

Package: GO Operational (query)

Task 1. Provide administrator privileges to the GO Operational (query) package for system administrators.

Because you can only specify security in the Publish wizard on the first publish of a package, this first step will remove the package from the Content Store which will reset the publish history. Alternately, you can just configure admin access for a package in the package properties.

1. Launch **IBM Cognos Connection**, log in, and then delete the **GO Operational (query)** package.
2. In **Framework Manager**, open **GO Operational.cpf** located in **C:\Edcognos\B5152\CBIFM-Start Files\Module 14\GO Operational**.

3. If prompted, log in as User ID **admin**, and Password **Education1!**.
4. In the **Project Viewer** pane, expand **Packages**, right-click **GO Operational (query)**, click **Publish Packages**, and then click **Next**.

The Add Security page is a quick and easy method of adding initial security to a package. As stated on the page, these settings are only available on the initial publish. After being published, you must modify permissions using the Permissions dialog box that will be demonstrated later in this demo.

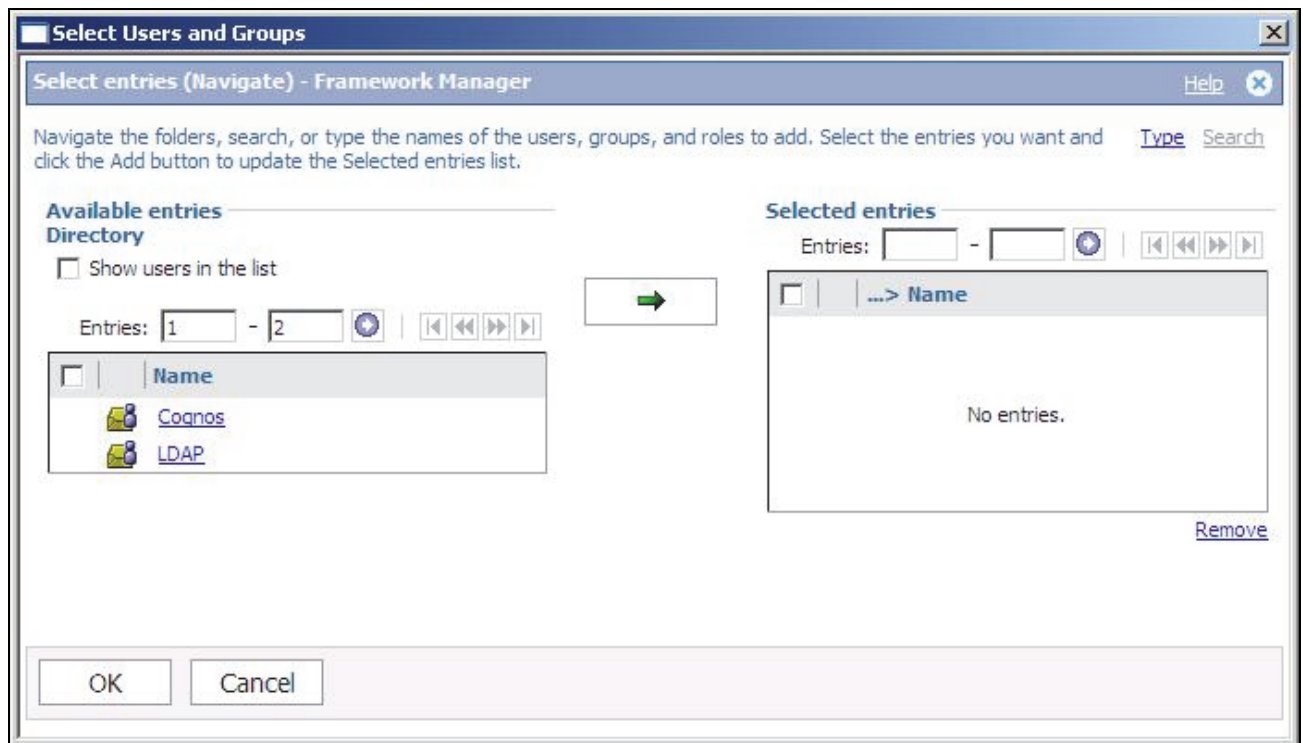
On the Add Security page, note that there are two tabs. You have the opportunity to grant either user access or administration access to specific security users, groups, or roles.

- The User tab grants Read, Write, Execute, and Traverse permissions.
- The Administrator tab grants Read, Write, Set Policy, and Traverse permissions. Set Policy gives the ability to modify security permissions.

You will specify security for both users and administrators. You will allow user access to the package for the Query Users role and administrator access to the System Administrators role to see the effects in IBM Cognos.

5. Click **Add**.

The results appear as follows:



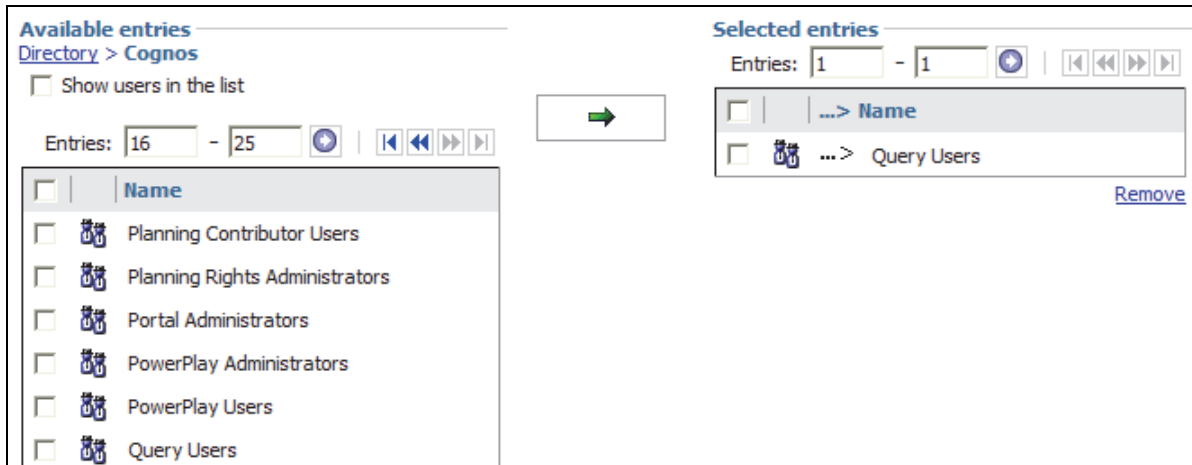
Note: This UI is the same as in IBM Cognos Connection. Framework Manager is simply accessing it.

There are two sources to identify security users, groups, and roles.



- 'Cognos' is the container for groups and roles defined within IBM Cognos.
- 'LDAP' is the container for users and groups defined within the Sun Java System Server Console.

6. Click **Cognos**, click **Next Page** , select **Query Users**, and then click **Add** .



The results appear as follows:



You can also select groups or individual users from the authentication provider (in this case Local LDAP) namespace to meet any security requirements you may have.

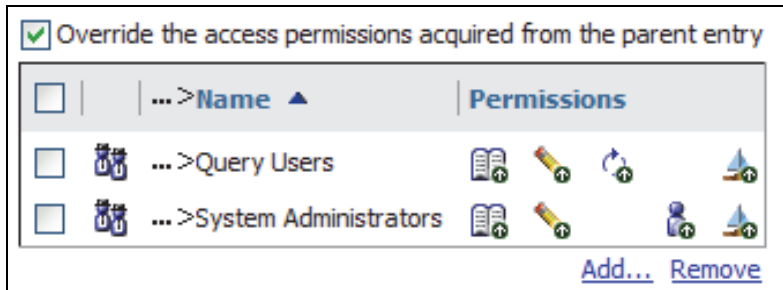
7. Click **OK**, click the **Administrator Access** tab, and then click **Add**.
8. Click **Cognos**, and then click **Next Page** .
9. Select **System Administrators**, and then click **Add** .
10. Click **OK**, click **Next**, and then deselect the **Verify the package before publishing** check box.
11. Click **Publish**, and then click **Finish**.

Task 2. Test the Package Security.

1. In **IBM Cognos Connection**, click **Refresh** , and then beside **GO Operational (query)**, click **Set Properties** .

- Click the **Permissions** tab.

The results appear as follows:



- View the pop-up tips over each of the **Permissions** icons.

You are looking at the permissions for all the defined users, groups and roles (in this case there are only roles defined).

- Query Users have Read, Write, Execute, and Traverse permissions
- System Administrators have Read, Write, Set Policy, and Traverse permissions


You can grant or deny different permissions to individual users, groups, and roles in this UI through IBM Cognos Connection or Framework Manager. You can also add or delete users, groups, and roles.

Note that in the first check box, Override the access permissions acquired from the parent entry, is selected. This is because you explicitly added roles, which Overrides the default setting of the parent container, Public Folders.

- Click **Cancel**, and then, on the title bar beside **Admin Person**, click **Log Off**.

You will now log on as a Query User member to see the effects of package access security.

- Log on as **whites** (Sally White) (password = **Education1!**).

6. Click **IBM Cognos content**, and then, beside **GO Operational (query)**, click **Set Properties** .

Sally White is only a member of the Query Users role and not the System Administrators role. Notice that there is no Permissions tab. You were able to see the package in Public Folders (and you can access it in the Studios), but you cannot view or edit its permissions without the Set Policy permission.

You will now log on as a user from the Authors role.

7. Click **Cancel**, and then, on the title bar beside **Sally White**, click **Log Off**.
8. Log on as **brettonf** (Frank Bretton) (password = **Education1!**).
9. Click **IBM Cognos content**.

Notice the GO Operational (query) package is not visible. This is because Frank Bretton is only a member of the Authors role that does not have user access defined for this package.

You will give the Authors role user access permissions and then limit the access.

Task 3. Edit the package security in Framework Manager.

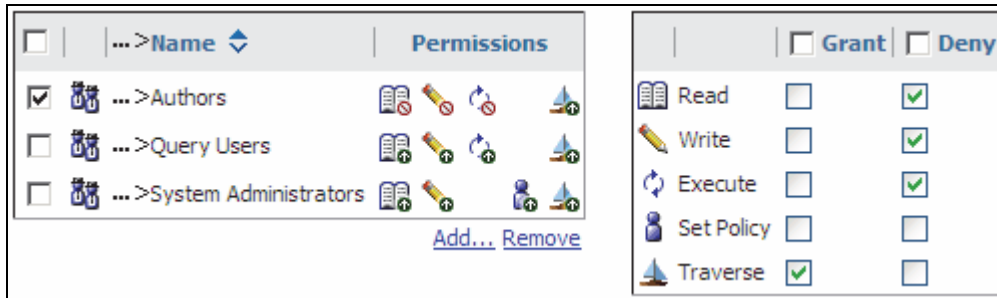
1. In **Framework Manager**, ensure the **GO Operational (query)** package is selected, and then from the **Actions** menu, point to **Package**, and then click **Edit Package Settings**.

2. Click the **Permissions** tab.

Again, this is the same page you saw in IBM Cognos Connection. You can use either one, as they both access the same settings in the portal. You have access to the Permissions tab because you are still logged on as admin within Framework Manager.

3. Click the **Add** link to add the **Authors** role from the **Cognos** namespace.

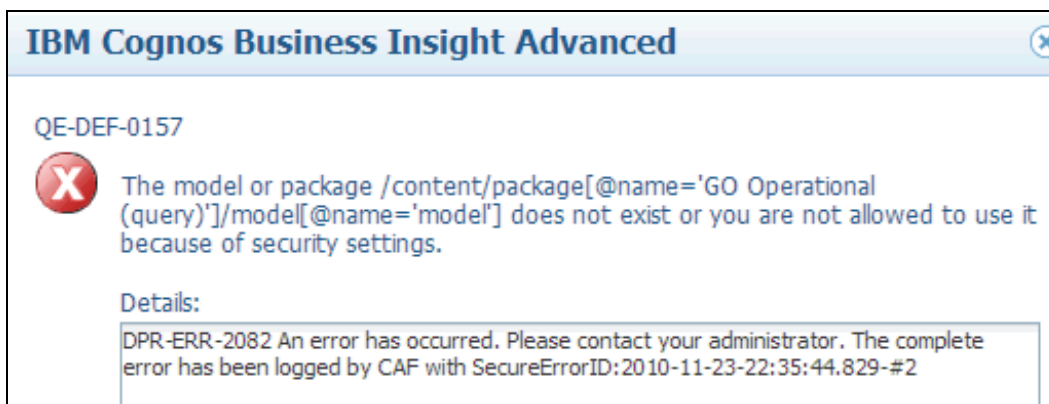
4. Select the **Authors** check box, and then specify permissions as shown below:



Notice that Read, Write, and Execute permissions have been explicitly denied, Set Policy has been implicitly denied, and Traverse has been granted.

5. Click **OK**, and then publish the **GO Operational (query)** package.
6. In **IBM Cognos Connection**, still logged on as **Frank Bretton**, click **Refresh**. The GO Operational (query) package now appears.
7. From the **Launch** menu, click **Business Insight Advanced**. Notice that the GO Operational (query) package is no longer a link.
8. Click **Cancel**, and then click **GO Operational (query)**.
9. While in the **GO Operational (query)**, from the **Launch** menu, click **Business Insight Advanced**.
10. Click **Create New**, and then double-click **List**.

The results appear as follows:



Frank Bretton cannot author reports with this package because he has been denied the permission.

11. Click **OK**, and then close Business Insight Advance.
12. In **Framework Manager**, from the **Actions** menu, point to **Package**, and then click **Edit Package Settings**.
13. Click the **Permissions** tab, select the **Authors** check box, and then select the **Grant** check box for the **Read**, **Write**, and **Execute** permissions.
14. Click **OK**, and then publish the package again.
15. In **IBM Cognos Connection**, still logged on as **Frank Bretton**, launch **Business Insight Advanced** selecting the **GO Operational (query)** package.
16. Click **Create new**, and then double-click **List** (you may need to log off and then log on again to clear the session cache and see the effects of the security change).

Business Insight Advanced opens and Frank Bretton has access to the contents of the package.
17. Close the browser, and then in Framework Manager, save the project.

Results:

You limited administrative control of the GO Operational package to members of the System Administrators role. You also allowed access to the package for members of the Query Users and Authors role. You tested various settings to see the effects in IBM Cognos Connection and in Query Studio.

Specify Data Security

- To specify data security:
 1. create a security filter to restrict the data returned by a query subject (for example, Retailer.Region='Italy')
 2. add the groups and roles to which the security filter will be applied (for example, the Italy group)
- This affects:
 - authors when they create a report
 - users when they run a report

A security filter controls the data that is shown to report authors when they test their reports, and when users run reports. The filter expression can incorporate macros, parameter maps, and session parameters.

Demo 2: Specify Data Security

Purpose:

Sales managers at The Great Outdoors Company want to ensure that Camping Equipment sales representatives only see orders relating to the Camping Equipment product line.

To accomplish this, you first need to create and add members to the Sales Managers and Camping Equipment Reps groups. You will grant these groups access to the GO Operational (query) package.


You will then apply a security filter to the Products query subject to restrict their access to camping equipment data. Finally, you will publish the package and view the results in Business Insight Advanced.

Components: Framework Manager, Business Insight Advanced

Project: GO Operational

Package: GO Operational (query)

Task 1. Create the Sales Managers and Camping Equipment Reps groups.

1. Launch **IBM Cognos Connection**, log on as **admin** (password = **Education1!**).
2. Under **Administration**, click **Administer IBM Cognos content**, and then click the **Security** tab.
3. Click **Cognos**, and then click **New Group** .
4. In the **Name** box, type **Camping Equipment Reps**, and then click **Next**.
5. Click **Add**, click **LDAP**, and then click **Search** in the top right corner.

6. Click **Advanced**, and then in the **Type** list, click **Users**.
7. In the **Find text in** box, type **kunze**, and then click **Search**.

The results appear as follows:


Available entries
 Directory > LDAP
 Find text in: Name field ▼

kunze Search Advanced

Method: Contains the exact string ▼ Type: Users ▼
 Scope: This folder and its subfolders ▼

Results: Entries: 1 - 1

<input type="checkbox"/>	...> Name
<input checked="" type="checkbox"/>	...> Jörg Kunze (kunzej)

8. Under **Results**, select the **Jörg Kunze** check box, click **Add** , and then click **OK**.
9. Click **Finish**.
10. Click **New Group**, and then repeat steps 3 to 9 to create the **Sales Managers** group and add **Kazumi Uragome (uragomek)** as a member.
11. Beside **Authors**, click **Set Properties**, and then click the **Members** tab.
12. Click **Add**, click **Cognos**, and then click **Next Page**.
13. Select **Sales Managers**, click **Add**, and then click **OK** twice.

Task 2. Grant access to the GO Operational (query) package.

1. In **Framework Manager**, ensure the **GO Operational (query)** package is selected.
2. From the **Actions** menu, point to **Package**, and then click **Edit Package Settings**.
3. Click the **Permissions** tab.
4. Click **Add**, click **Cognos** and then select **Camping Equipment Reps** and **Sales Managers** groups you just created.
5. Click **Add**, and then click **OK**.
6. Select the **Camping Equipment Reps** and **Sales Managers** check boxes.
7. Under **Grant** in the right box, select the **Read**, **Write**, **Execute**, and **Traverse** check boxes.

The results appear as follows:

☒ Override the access permissions acquired from the parent entry

<input type="checkbox"/>	...>Name ▲	Permissions
<input type="checkbox"/>	...>Authors	
<input checked="" type="checkbox"/>	...>Camping Equipment Reps	
<input type="checkbox"/>	...>Query Users	
<input checked="" type="checkbox"/>	...>Sales Managers	
<input type="checkbox"/>	...>System Administrators	

[Add...](#) [Remove](#)

	<input type="checkbox"/> Grant	<input type="checkbox"/> Deny
Read	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Write	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Execute	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Set Policy	<input type="checkbox"/>	<input type="checkbox"/>
Traverse	<input checked="" type="checkbox"/>	<input type="checkbox"/>

8. Click **OK**.

Task 3. Create a security filter for the Product query subject.

1. In the **Project Viewer** pane, expand **GO Operational Model>Consolidation View**.
2. Click **Products**, and then from the **Actions** menu, click **Specify Data Security**.
3. Click **Add Groups**.
4. Click **Cognos**, select the **Camping Equipment Reps** check box, and then click **Add**.
5. Click **OK**.

At this point, if you already had another group defined with a filter on camping equipment products, you could click below Based On and specify that other group. In this situation you do not, so you will create a new filter.

6. Click the box in the first row under the **Filter** column, and then click **Create/Edit Embedded**.
7. In the **Available Components** pane, expand **GO Operational Model>Consolidation View>Products>Codes**.
8. Double-click **Product Line Code**, at the end of the expression, type = **991**.

The results appear as follows:



This filter will ensure that members of the Camping Equipment Reps will only see camping equipment products.

9. Click **OK** twice, and then save the project.

Task 4. Publish the package and view the results in Business Insight Advanced.

1. Publish the **GO Operational (query)** package.
2. In **IBM Cognos Connection**, log off, and then log on again with as **uragomek** (password = **Education1!**).


Uragomek is a member of the Sales Managers group.

3. Launch **Business Insight Advanced** selecting the **GO Operational (query)** package to create a **List** report.
4. In the **Insert Data** menu, expand **Sales (query)>Products**, and then double-click **Product Line** and **Product Type**.

The results appear similar to the following:

Product line	Product type
Personal Accessories	Binoculars
Mountaineering Equipment	Climbing Accessories
Camping Equipment	Cooking Gear
Personal Accessories	Eyewear
Outdoor Protection	First Aid
Golf Equipment	Golf Accessories
Outdoor Protection	Insect Repellents
Golf Equipment	Irons
Personal Accessories	Knives
Camping Equipment	Lanterns
Personal Accessories	Navigation
Camping Equipment	Packs
Golf Equipment	Putters
Mountaineering Equipment	Rope

Notice that Kazumi Uragome can see product data for all product lines. He is a member of the Sales Managers group and the security filter applied in Task 2 does not apply to this group.

5. On the toolbar, click **Save** , in the **Name** box, type **Demo 2 - Security Filter Test**, and then click **Save**.
6. Close IBM Cognos Business Insight Advanced, log off, and then log on again as **kunzej** (password = **Education1!**).

Jörg Kunze is a member of the Camping Equipment Reps group.

7. Click **IBM Cognos content**, click the **GO Operational (query)** folder, and then click **Demo 2 - Security Filter Test**.

The result appears as shown below:

Product Line	Product Type
Camping Equipment	Cooking Gear
Camping Equipment	Lanterns
Camping Equipment	Packs
Camping Equipment	Sleeping Bags
Camping Equipment	Tents

Due to the data security placed on the Product query subject for the Camping Equipment Reps group (of which Jörg Kunze is a member), only product quantities from the Camping Equipment product line are shown in the report.

8. Close the browser.

Note: If a user does not belong to any of the groups specified in security filters, then they will have unrestricted access to the data, regardless of the restrictions specified by the filter expressions. In some cases, you may want to avoid this scenario and completely restrict access to particular groups or roles.

To completely restrict access, you can create a filter expression that will always resolve to a false outcome. For example, you can create a filter expression that reads [Consolidation View].[Products].[Product Line Code] = 1 where 1 does not exist in the data for Product Line Code. This expression will always resolve to false. You could then specify the groups or roles to which this filter expression will be applied. The result is that a user who is a member of the defined groups or roles for this filter expression will be completely denied access to the data (provided they are not members of a group that does have access to the data).

Results:

You created and added members to the Sales Managers and Camping Equipment Reps groups. You granted these groups access to the GO Operational (query) package. You then applied a security filter to the Products query subject to restrict access to product line data. Finally, you published the package and viewed the results in Business Insight Advanced.

Specify Object Security

- Allow or deny access to objects such as:
 - namespaces
 - folders
 - query subjects
 - query items
 - filters

When granting access to objects, ensure that the selected users, groups, or roles have access to the package that contains them.

An example of applying object security is to only allow access to the Sales Targets namespace to the Sales Managers group. Then only members of the Sales Manager group will see the Sales Target namespace in the studios.

Examine Object Security Rules

- With no object security applied, all objects are accessible.
- After applying object security, only those objects to which access has been granted will be accessible by the selected users, groups, or roles.
- Child objects inherit the security of parent objects.
- Deny overrides Allow.

When you begin to apply object security (Allow or Deny), all other objects, other than the children of the object you specified security on, are automatically not visible to any user (including members of the System Administrator role) until access is explicitly granted. Only those objects to which access has been explicitly granted for the selected users, groups, or roles, are visible.

When you set object security on a parent object, a child object inherits the security of the parent if object security has not already been specified for the child object.

In the case of an access conflict (for example, being a member of two groups with conflicting access to an object), denied access to the object overrides granted access to it.

Specify Object Security: Methodologies

- There are two approaches when beginning to implement object-based security in a model:
 - Allow access to all objects, and then restrict
 - Restrict an object, and then allow access as required

Allow access to all objects, and then restrict:

1. Make all objects accessible to everyone (open the root namespace's access control list, and then select the Allow check box for the Everyone group).
2. Restrict access to specific child objects for selected users, groups, or roles by either explicitly denying access (Deny option) or implicitly denying access by leaving both Allow and Deny options unselected. Deny overrides allow which ensures that a user does not accidentally gain access to an object through another group or role that has access.

Restrict an object, and then allow access as required:

1. Apply initial security to an object to allow specific users, groups or roles access to the object (this automatically implements security across the model and makes all other objects inaccessible to all users).
2. Grant access to specific objects for selected users, groups, or roles as required.

Demo 3: Specify Object Security

Components: IBM Cognos Connection, Framework Manager, Business Insight Advanced

Project: GO Operational

Package: GO Operational (query)

Purpose:

You want members of the Sales Managers group to access all metadata under the GO Operational Sales (query) namespace. This includes access to the Sales Target namespace, which will allow them to run reports that include the projected sales targets for each sales rep.

At the same time, you want to restrict the access to the Sales Targets (query) namespace so that members of the Sales Reps group cannot access projected sales targets for each sales rep.

You will begin by creating and adding members to a new group called Sales Reps.

You will grant access to the GO Operational (query) package to the new Sales Reps group, and grant or deny access to the appropriate objects for the Sales Managers and Sales Reps groups.

Lastly, you will publish the GO Operational (query) package and view the results in Business Insight Advanced.

Task 1. Create a new Sales Reps group and add a member.

1. Launch **IBM Cognos Connection**, log on as **admin** (password = **Education1!**).
2. Under **Administration**, click **Administer IBM Cognos content**, and then click the **Security** tab.
3. In the **Cognos** namespace, create a new group called **Sales Reps**, and then add **Bart Scott** and **Daniel Turpin** from **LDAP**.

Task 2. Grant access to the GO Operational (query) package.

1. In **Framework Manager**, in the **Project Viewer** pane, select the **GO Operational (query)** package, and then from the **Actions** menu, point to **Package**, and then click **Edit Package Settings**.
2. Click the **Permissions** tab, click **Add**, and then click **Cognos**.
3. Click **Next Page**, select the **Sales Reps** check box, and then click **Add**.
4. Click **OK**, select the **Sales Reps** check box, and then under **Grant**, select the **Read**, **Write**, **Execute**, and **Traverse** check boxes.

The results appear as follows:

☒ Override the access permissions acquired from the parent entry

<input type="checkbox"/>	Name	Permissions
<input type="checkbox"/>	...>Authors	Read, Write, Execute, Traverse
<input type="checkbox"/>	...>Camping Equipment Reps	Read, Write, Execute, Traverse
<input type="checkbox"/>	...>Query Users	Read, Write, Execute, Traverse
<input type="checkbox"/>	...>Sales Managers	Read, Write, Execute, Traverse
<input checked="" type="checkbox"/>	...>Sales Reps	Read, Write, Execute, Traverse
<input type="checkbox"/>	...>System Administrators	Read, Write, Execute, Traverse

[Add...](#) [Remove](#)

	Grant	Deny
Read	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Write	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Execute	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Set Policy	<input type="checkbox"/>	<input type="checkbox"/>
Traverse	<input checked="" type="checkbox"/>	<input type="checkbox"/>

5. Click **OK**.

Task 3. Specify object security on the GO Operational Sales (query) namespace for the Sales Managers and Sales Reps groups.

1. In **Project Viewer**, under **Presentation View**, click the **GO Operational Sales (query)** namespace.
2. From the **Actions** menu, click **Specify Object Security**, and then click **Add**.
3. Click **Cognos**, click **Next Page**, and then select the **Sales Managers** and **Sales Reps** check boxes.
4. Click **Add**, and then click **OK**.
5. Under **Allow**, select the **Sales Managers** and **Sales Reps** check boxes, and then click **OK**.

A message appears indicating that for an object to be visible in IBM Cognos, it must be made visible to the user within the package and by allowing the user access to the object.

6. Click **OK**.

The security you just specified on the GO Operational Sales (query) namespace will be inherited by all its children.

You will now restrict access to the Sales Targets (query) namespace.

Task 4. Specify object security on the Sales Targets (query) namespace for the Sales Reps group.

1. Expand the **GO Operational Sales (query)** namespace, click the **Sales Targets (query)** namespace, and then from the **Actions** menu, click **Specify Object Security**.

The Specify Object Security box appears. Notice that the Sales target namespace has inherited the security that was applied to its parent object, the GO Operational Sales (query) namespace.

2. Beside **Sales Reps**, select the **Deny** check box, and then click **OK**.

A message appears, indicating that the security for the Sales Targets (query) namespace will override the settings already specified for the GO Operational Sales (query) namespace. Rather than select deny, you could also just have deselected the Allow setting for Sales Reps, which would have implicitly denied access, but the method you have implemented ensures members of this group will not have access regardless of any other group they belong to.

3. Click **OK**, and then save the project.

Task 5. View the results in Business Insight Advanced.

1. Publish the **GO Operational (query)** package.
2. In **IBM Cognos Connection**, log off, and then log on again as **uragomek** (password = **Education1!**).

3. Under **My Actions**, click **Author business reports**, and then click **GO Operational (query)**.

This user, being a member of the Sales Managers group, has access to all namespaces in the package, including the Sales Targets (query) namespace. This is due to the security that you specified.

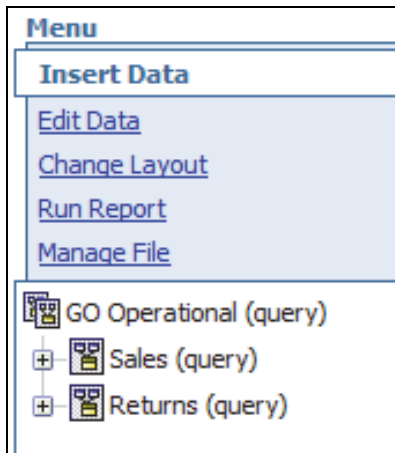
4. Click **Create new**, and then double-click **List**.
5. In the right pane, expand all of the namespaces.

Notice that they are empty. There are no query subjects available.

You will investigate this issue shortly in Framework Manager.

6. Close **IBM Cognos Business Insight Advanced**.
7. Click **Log off**, and then log on as **turpind** (Daniel Turpin) (password = **Education1!**).
8. Click **Query my data**, and then click **GO Operational (query)**.

The results appear as follows:



This user, being a member of the Sales Reps group, does not have access to the Sales target (query) namespace. This is due to the object security that you specified. This group has been explicitly denied access to this object.

9. Expand any of the namespaces.
Again, no objects are available.
10. Close the browser.

Task 6. Grant access to objects pointed to by shortcuts.




1. In **Framework Manager**, in the **Project Viewer**, expand **GO Operational Sales (query)>Sales (query)**.

When you granted access to the GO Operational Sales (query) namespace, the security is inherited at all levels below it, which is why you see the namespaces. But the namespaces do not contain any query subjects, only shortcuts to query subjects located elsewhere in the project. In this case they are located in the Consolidation View. Security on shortcuts is not inherited by the underlying objects. Even though users have access to the shortcuts, they still need access to the underlying objects in order to see them.

You will grant access to everyone for all other objects in the model that have not already been configured with object security. This way you can ensure that users will have access to underlying objects and then restrict as required.

2. Click the **GO Operational Model** namespace (the root namespace), and then from the **Actions** menu, click **Specify Object Security**.

The results appear as follows:

	Allow	Deny
 Everyone [Directory > Cognos]	<input type="checkbox"/>	<input checked="" type="checkbox"/>
 Sales Managers [Directory > Cognos]	<input type="checkbox"/>	<input type="checkbox"/>
 Sales Reps [Directory > Cognos]	<input type="checkbox"/>	<input type="checkbox"/>

Notice that no one is allowed access. This applies to all other objects you have not yet configured object security for. By allowing access to everyone, all objects will inherit this setting with the exception of the objects you already configured.

3. Under **Allow**, select the **Everyone** check box, and then click **OK**.

You may also choose to use another group to open up access to users such as the All Authenticated Users group, or any other group that meets your requirements. Using the Everyone group is just used for the purposes of this demo and not a requirement for this method to work.

4. Save the project, and then publish the **GO Operational (query)** package.
5. Launch **IBM Cognos Connection**, log on as **turpind** (password = **Education1!**).
6. Launch **Business Insight Advanced** selecting the **GO Operational (query)** package for a **List** report.

Daniel Turpin still cannot see the Sales Targets (query) namespace even though he has access to the underlying query subjects. This is because he has been denied access to the namespace and the shortcuts it contains.


Also, notice that the **Retailer Location Filters** folder is now visible. This is also a shortcut in the **GO Operational Sales (query)** namespace. Now that the underlying object is available to everyone, it is visible.

7. Expand **Sales (query)>Sales Fact**.
You now see the query subjects and query items.
8. Close the browser.

Results:

You granted access to the GO Operational (query) package to the new Sales Reps group. You granted access to the GO Operational Sales (query) namespace for the Sales Managers and Sales Reps groups. You then denied access to the Sales Targets (query) namespace for the Sales Reps group.

After publishing the GO Operational (query) package and viewing the results in Query Studio, you identified an issue with your security implementation. You then opened up security on all remaining objects in the project at the root namespace to fix the problem.

Business Analytics


Create Dynamic Data Security Using a Macro

Filter Expression

```
[Sales Target Fact].[Sales Staff Code] =
#$$SecurityLookup{$account.defaultName}#
```


Parameter Map

Key	Value
Bart Scott	60
Ana Orozco	80
Alex Rodriguez	53

Session Parameter

Generated SQL

```
.....
Where
Sales_Target_Fact.Sales_Staff_Code = 60
```



© 2010 IBM Corporation

Here is an additional and advanced technique you can use to apply security.

Use environment session parameters in a macro to create a dynamic data security filter when you want the security to only apply to specific groups.

You could also use the macro in the slide example above directly in the query subject, but then the security would be applied to everyone using the query subject rather than specific groups or people. For example, you want members of the Sales Managers group to have access to all sales targets and not be restricted. Conversely, you want to restrict members of the Sales Reps group to only have access to their own data and not other sales reps. Data security filters can accomplish this.

Demo 4: Implement Security Using a Macro in a Data Security Filter

Components: Framework Manager, Business Insight Advanced

Project: GO Operational

Package: GO Operational (query)

Purpose:

You have been asked to publish a package in which members of the Sales Managers group can view sales targets for all employees, but members of the Sales Reps group can only see their own sales targets.

You will create a parameter map, and a macro that references an environment session parameter to implement data security for a particular group on a query subject.

Task 1. Change the security from the last demo.

Because you are allowing Sales Reps members to have access to their own sales targets data, you will need to alter the security you previously implemented and allow Sales Reps members access to the Sales Targets (query) namespace.

1. In **Framework Manager**, in the **Project Viewer** pane, under **GO Operational Sales (query)**, click the **Sales Targets (query)** namespace.
2. From the **Actions** menu, click **Specify Object Security**.
3. Select **Allow** for **Sales Reps**, and then click **OK**.

Task 2. Create a parameter map.

1. In the **Project Viewer** pane, right-click **Parameter Maps**, point to **Create**, and then click **Parameter Map**.
2. In the **Name** box, type **SecurityLookup**, select **Base the parameter map on existing Query Items**, and then click **Next**.
3. Expand **Consolidation View>Staff by Location**, click **Staff Full Name**, and then click **Set as Key**.
4. Expand the **Codes** folder, select **Sales Staff Code**, and then click **Set as Value**.

Staff Full Name acts as the unique key for the parameter map and Sales Staff Code acts as the substitution value that will be used to apply security in a data security filter.

By basing this parameter map on existing query items in the data source, the parameter map will always reflect the latest data updates. As new employees join the company, they will automatically be reflected in the parameter map.

Again, to optimize this approach, you can filter the query subject used to feed the parameter map to make it more efficient. In this case, first make a copy of the query subject, filter it dynamically on Staff Full Name = account.defaultName, and then base the parameter map off new query subject. It will only return one record when the parameter map is called and prevents scanning the whole table. Since this is a relatively small table and in the interest of time, you will simply use the existing query subject.

5. Click **Next**.

The results appear as follows:

Default value:	
Key	Value
Denis Pagé	10004
Élizabeth Michel	10005
Émile Clermont	10006
Étienne Jauvin	10007
Elsbeth Wiesinger	10012
Else Mönke	10013
Frank Fuhlroth	10014
Gunter Erler	10015
Björn Winkler	10016
Fritz Hirsch	10017
Jörg Kunze	10018
Silvano Allessori	10019
Maria Iacobucci	10020
Alessandra Torta	10021
Belinda Jansen-Velasquez	10022
Ellen Shapiro	10023
Jan Haverkamp	10024
Kick Kalkman	10025

You can provide a default value in cases where no match is found in the parameter map. In these cases, the default value will be used, which will return specific data of your choice. In this case however, you do not want to return any data if there is no match so you will leave the field blank.

6. Click **Finish**.


Task 3. Apply data security to a query subject.

You will now secure the original SALES_TARGET query subject in the Foundation Objects View so that the security will be carried forward to all objects that reference it, now and in the future.

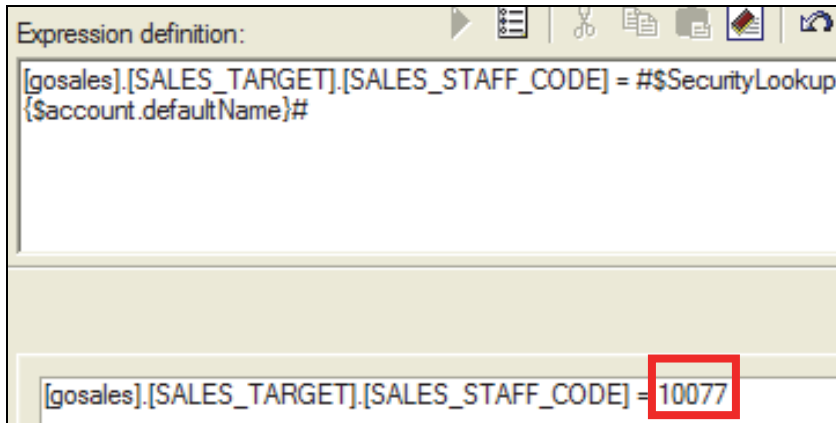
1. Under **Foundation Objects View**>**gosales**, click **SALES_TARGET**.
2. From the **Action** menu, click **Specify Data Security**.
3. Click **Add Groups**, click **Cognos**, and then click **Next Page**.
4. Add **Sales Reps**.
5. Click **OK**, and then, under the **Filter** column, select **Create/Edit Embedded**.
6. Under **Available Components**, expand **GO Operational Model**>**Foundation Objects View**>**gosales**>**SALES_TARGET**.
7. Double-click **SALES_STAFF_CODE** to add it to the expression definition.
8. At the end of the expression, type **=**.
9. Under **Available Components**, click the **Parameters** tab, expand **Parameter Maps**, and then double-click **SecurityLookup** to add it to the expression.
10. In the **Expression definition**, place the cursor between **}** of the **#\$SecurityLookup{}** expression.
11. Under **Available Components**, collapse **SecurityLookup**, expand **Session Parameters**, and then double-click **account.defaultName** to add it to the expression.

The Expression definition indicates that there is an error with the expression represented by a red squiggly underline. The pane below describes the error. This is due to the fact that you are currently logged in as admin. This account name does not exist in the data and therefore can not resolve the filter.

You will now override the account.defaultName value in order to test and validate this filter.

12. Click **Options** , under **Session parameter**, click **Set**, and then in the **Override Value** field for **account.defaultName**, type **Daniel Turpin**.
13. Click **OK**, and then click **OK** again.

The results appear as follows:



Notice that the red line is gone and that the expression now resolves to a value.

14. Click **OK**, and then click **OK** again.
15. Save the project.

Task 4. Publish and Test the package in Business Insight Advanced.

1. Publish the **GO Operational (query)** package.
2. Open **IBM Cognos Connection**, log on as **turpind** (password = **Education1!**).

Remember, Daniel Turpin is a member of the Sales Reps group.

3. Launch **Business Insight Advanced** selecting the **GO Operational (query)** package.

4. Click **Create new**, and then double-click **List**.
5. In the **Insertable Objects** pane, expand **Sales Targets (query)>Staff by Location**.

6. Add **Staff Full Name** to the report.

All names are returned since the filter is on the sales target data.

7. Expand **Sales Target Fact**, and then add **Sales Target** to the report.

The results appear as follows:

Staff Full Name	Sales Target
Daniel Turpin	30,808,000
Overall - Summary	30,808,000

The report is now limited to sales target for Daniel Turpin.

8. Save the report as **Demo 4 - Security Filter Test**.
 9. Log off, and then log on as **scottb** (password=**Education1!**).
- Bart Scott is also a member of the Sales Reps group.
10. Click **IBM Cognos content**, click **GO Operational (query)**, and then click **Demo 4 - Security Filter Test**.

The results appear as follows:

Staff Full Name	Sales Target
Bart Scott	38,209,900
Overall - Summary	38,209,900

Data is now limited to this user.

You will now log on as a member of the Sales Managers group to test their access.

11. Log off, and then log on as **uragomek** (password=**Education1!**).

12. Run the **Demo 4 - Security Filter Test** report.

The results appear as follows:

Staff Full Name	Sales Target
Aaltje Hansen	22,118,800
Abram Ruiz	44,933,900
Adda Heijman	21,862,800
Adriaantje Haanraads	24,468,300
Agatha Reyes	21,327,200
Agnelo Chavez	15,469,300
Agnes Ramos	25,934,800
Aidan Chaplin	10,978,300
Aiko Watanabe	39,354,200
Aila Forssell	18,515,400
Aimi Tanaka	14,553,010
Akemi Takahashi	43,558,700

The report returns all employees and their sales targets.

13. Close the browser.

Results:

You published a package in which members of the Sales Managers group can view sales targets for all employees, but members of the Sales Reps group can only see their own sales targets.

You created a parameter map, and a macro that references an environment session parameter to implement data security for the Sales Reps group on the SALES_TARGET query subject.

Demo 5: Remove Security

Purpose:

In order to remove the impact of security applied in this module on future modules, you will remove the security applied throughout this module. This is a good exercise as it will show you how to undo various security implementations.

Components: **Framework Manager**

Project: **GO Operational**

Package: **GO Operational (query)**

Task 1. Remove security from the package.

1. Select the **GO Operational (query)** package, and then go to **Actions>Package>Edit Package Settings**.
2. Click the **Permissions** tab, and then clear the **Override the access permissions acquired from the parent entry** check box.

A window opens explaining that this will cause the parent's policies to be acquired.

3. Click **OK**.

The results appear as follows:

☐ Override the access permissions acquired from the parent entry

<input type="checkbox"/>	...>Name ▲	Permissions
	...>Analysis Users	
	...>Authors	
	...>Consumers	
	...>Controller Administrators	
	...>Controller Users	
	...>Data Manager Authors	
	...>Express Authors	
	...>Metrics Administrators	
	...>Metrics Authors	
	...>Metrics Users	
	...>Planning Contributor Users	
	...>Planning Rights Administrators	
	...>PowerPlay Administrators	
	...>PowerPlay Users	
	...>Query Users	
	...>Readers	
	...>Report Administrators	

This package is now available to all the roles that have access to the other packages found in the Public Folders area of IBM Cognos Connection.

4. Click **OK**.

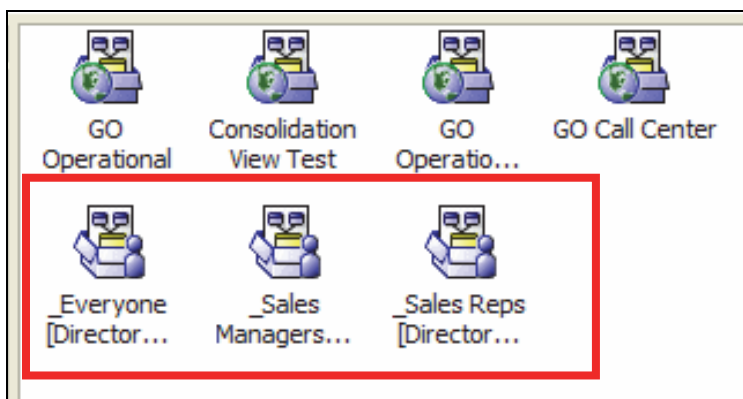
Task 2. Remove Data Security.

1. In the **Project Viewer**, under **Consolidation View**, select **Products**.
2. From the **Actions** menu, click **Specify Data Security**.
3. Delete the **Camping Equipment Reps** group, and then click **OK**.
4. Repeat steps 1 to 3 to delete the group from **SALES_TARGET** in the **gosales** namespace.

Task 3. Remove object security.

1. In the middle pane, click **Explorer**.
2. In the **Project Viewer**, double-click the **Packages** folder to give it focus in the **Explorer**.

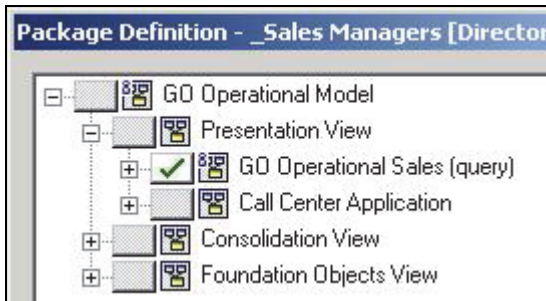
The results appear as follows:



Notice the security based packages. These packages contain the security information you specified earlier.

3. Double-click **_Sales Managers** to view the definition.

The results appear as follows:



Here you can see the objects this group has access to.

4. Click **Cancel**.

You will now remove all object security by deleting the **_Everyone** package.

5. Select the **_Everyone** package and then press delete.

A message appears stating that this role is required for security implementation and that removing it will remove all object security from the model. If you just wanted to remove the Sales Managers group security, you could just delete that one object.

6. Click **OK**, and then save and close the project.

Results:

You removed the user access and admin access restrictions you applied earlier by inheriting the GO Operational (query) parent's permissions setting.

You removed data security filters that you applied and you also removed all object security by deleting the **_Everyone role security package which removed all object security from the project.**

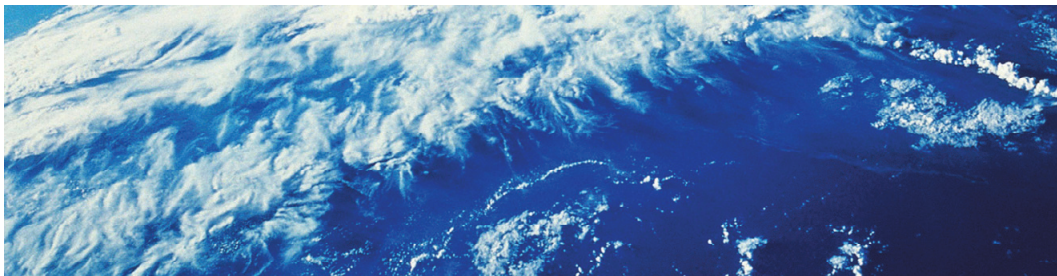
Summary

- You should now be able to:
 - navigate the IBM Cognos security environment
 - restrict access to packages
 - create and apply security filters
 - restrict access to objects in the model



Create Analysis Objects

IBM Cognos BI

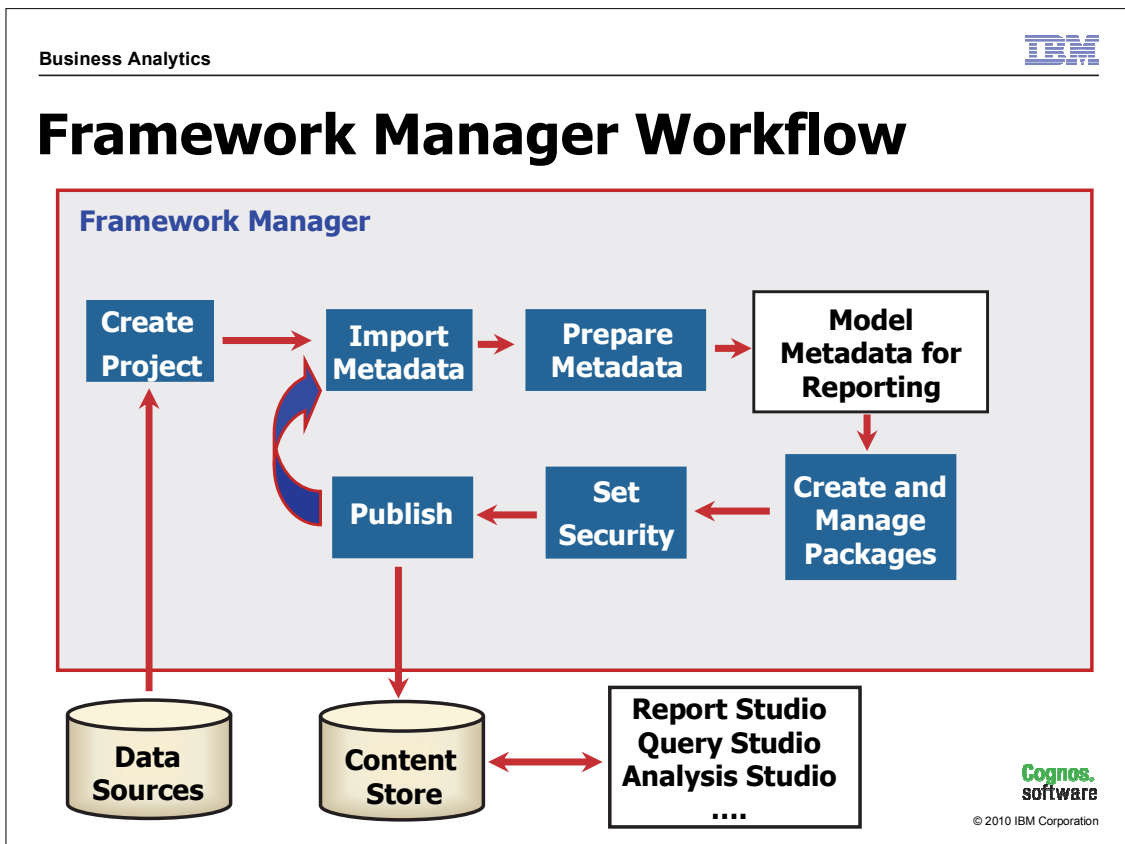


Business Analytics

© 2010 IBM Corporation

Objectives

- At the end of this module, you should be able to:
 - apply dimensional information to relational metadata to enable OLAP-style queries
 - sort members for presentation and predictability



This module deals with creating analysis objects that allow authors to perform OLAP-style queries.

Define Dimensionally Modeled Relational (DMR) Metadata

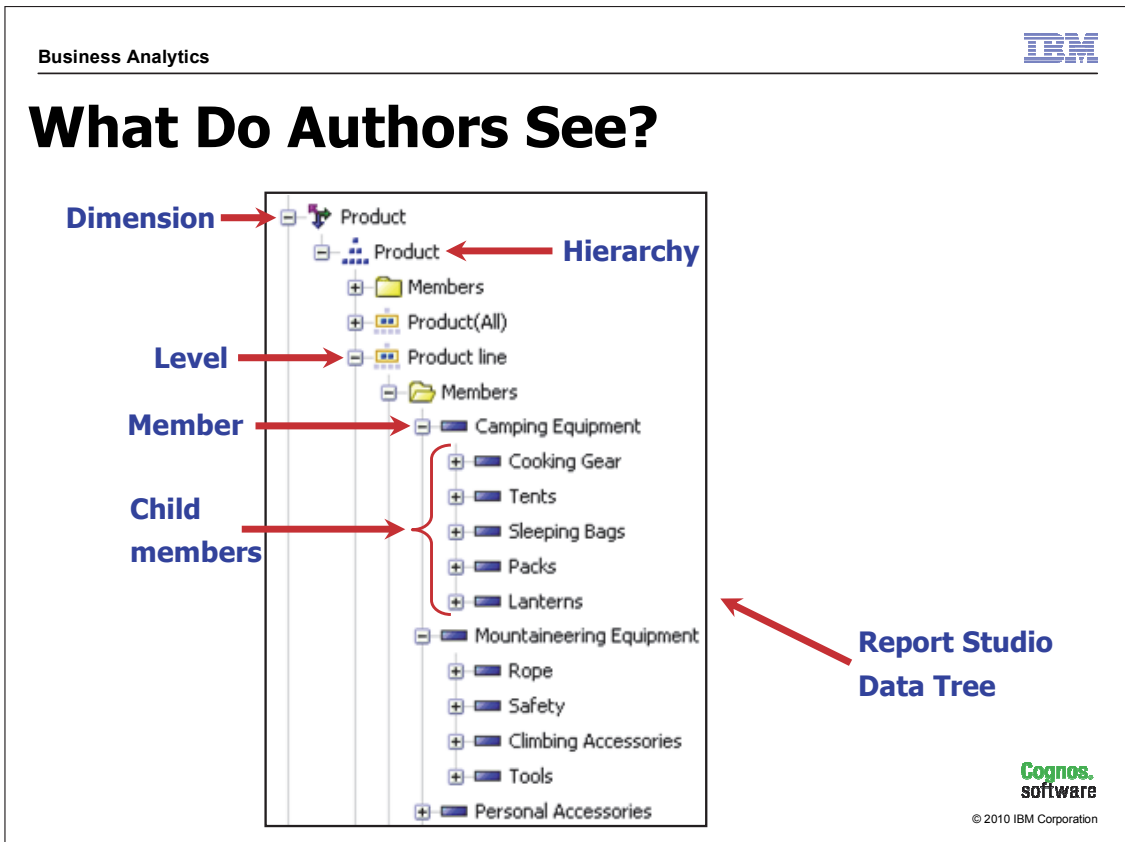
- In IBM Cognos, DMR refers to the dimensional information a modeler supplies for a relational data source to allow for OLAP-style queries.
- This information is defined through:
 - Regular Dimensions
 - Measure Dimensions
 - Scope Relationships

A dimensionally modeled layer can be applied to any metadata in star schema format.

When your metadata is in star schema format, you can provide hierarchy information to dimensions and measure scope.

DMR allows for OLAP-style queries that include drill up and down functionality and access to dimensional functions such as *parent*(*[member]*) or *except*(<Level>, *set*(*[member1]*, *[member2]*, *[member3]*)).

Again, data access strategies should be considered before embarking on any modeling solution. OLAP cubes or a relational star schema database that employs aggregate tables can yield better performance than providing dimensional information for an operational system.



Just as with OLAP data sources, authors will be presented with multi-dimensional metadata in the studios when you apply dimensional information to your model. They will also see members in member aware studios (Analysis Studio and Report Studio).

Package Types and IBM Cognos Studios

- Business Insight Advanced, Report Studio, Query Studio, and Event Studio can access all package types (Relational, OLAP, or DMR).
- Analysis Studio can only access dimensional packages (OLAP or DMR).

Business Insight Advanced, Report Studio, Query Studio, and Event Studio can access all types of packages. While Business Insight Advanced, Report Studio, and Event Studio are member aware (allow you to work directly with members), Query Studio is not.

Analysis Studio deals only with dimensional packages. If the package is not dimensional, then it will not be available when opening this studio.



Decide on a Modeling Style

Recommendation #9

- Use standard relational modeling to:
 - enable basic relational ad hoc querying and reporting
- Use DMR when you want to:
 - enable analysis on relational data within Analysis Studio
 - enable drill up and down functionality in reports and ad hoc queries
 - access member functions in the authoring tools

Cognos.
software

© 2010 IBM Corporation

Define Regular Dimensions

- Consists of one or more user-defined hierarchies
- Each hierarchy consists of
 - levels
 - keys
 - captions
 - attributes

Name	Role	Source
Product Name	_memberCaption	Products.Product Na ...
Product Number	_businessKey	Products.Codes.Prok ...
Product Description	_memberDescription	Products.Product De ...
Product Image		Products.Product Im ...
Introduction Date		Products.Introduction ...

Cognos.
software

© 2010 IBM Corporation

Hierarchies consist of levels, keys, captions, and attributes. Level information is used to roll up measures accurately when performing queries or analyses. Regular dimensions require that each level have key and captions specified, and that captions be of the type string. These items are used to generate members in the studio data trees (where applicable) and retrieve the members at run time.

To indicate that the keys in the levels above the current level are not necessary to identify the members in a level, select the Unique Level check box.

If you have a star schema that is in its final reporting form, and your requirement for the final model is DMR, you can specify determinants as required and then use the Convert to Regular Dimension and Convert to Measure Dimension features to quickly create a publishable model.

Differentiate Determinants and Regular Dimensions

Determinants:

- are specified for query subjects
- are required for dimensions with levels of granularity that have repeating keys
- are required for blobs
- do not provide OLAP functionality

Regular Dimensions:

- include hierarchies which are specified to allow OLAP-style analysis against relational data
- define levels for aggregation rollup
- are required for Analysis Studio

Determinants allow IBM Cognos to generate the appropriate SQL to aggregate facts appropriately (prevent double-counting).

Specifying determinants does not provide OLAP functionality or the ability to use the metadata in Analysis Studio.

Regular dimensions specify levels of a hierarchy, but do not control SQL generation with respect to applying a group by on keys that repeat for a particular level of granularity in the data. The exception to this rule is if you convert a data source query subject to a regular dimension. In this case the regular dimension uses joins and the defined levels to handle granularity. This means that the hierarchy information is used to define the navigation of a hierarchy as well as determinants.

Demo 1: Create Regular Dimensions

Purpose:

You have been asked to provide a business view for authors and business analysts that will allow them to author OLAP-style queries in all the authoring studios. You will accomplish this by using DMR techniques.

Their requirements are to analyze sales and sales target measures against products and time. You will begin preparing this business view by creating the required regular dimensions.

Component: Framework Manager

Project: GO Operational

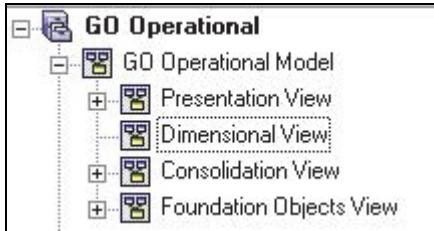
Task 1. Create the top level of a regular dimension for products.

You will organize dimensional objects in the Dimensional View namespace. Once the regular and measure dimensions are complete, you can use star schema groupings to populate the Presentation View.

1. In **Framework Manager**, close any projects that may be open, and then open the **GO Operational** project located at **C:\Edcognos\B5152\CBI-FM-Start Files\Module 15\GO Operational**.
2. If necessary, log in as User ID **admin**, and Password **Education1!**.
3. In the **Project Viewer** pane, create a new namespace under **GO Operational Model** called **Dimensional View**.

4. Drag the new namespace above the **Consolidation View** namespace.

The results appear as follows:



5. Right-click **Dimensional View**, point to **Create**, and then click **Regular Dimension**.
6. In the **Available items** pane, expand **Consolidation View>Products**.
7. Drag **Product Line** into the **Hierarchies** pane, right-click the top **Product Line** in the hierarchy column, and then click **Rename**.
8. Type **Products**, and then press **Enter**.
9. Rename **Product Line(All)** to **Product (All)**.
10. In the **Hierarchies** pane, click the **Product Line** level, in the **Available items** pane, expand the **Codes** query item folder, and then drag **Product Line Code** to the bottom right pane ("Select a level...") below Product Line.

You are prompted to select a role.

11. Select **_businessKey**.

Task 2. Create remaining levels for the Products regular dimension.

1. Drag **Product Type** below **Product Line** in the **Hierarchies** pane.
2. Drag **Product Type Code** to the bottom right pane, select **_businessKey** as the role, and then click the **Unique Level** check box above.

Selecting Unique Level indicates to IBM Cognos that this level does not require the level above it for uniqueness.

3. Drag **Product Name** below **Product Type** in the **Hierarchies** pane, and then rename this level to **Product**.
4. Drag **Product Number** to the bottom right panel and set it as the **_businessKey**.
5. Drag **Product Description** to the bottom right pane and set it as **_memberDescription**.
6. Click **Product Image**, Shift+click **Gross Margin**, and then drag the selected items to the bottom right pane.

The Product level is the lowest level of the hierarchy and is also represented by a unique key, in this case, Product Number. You will specify this level as unique as well.

7. Select the **Unique Level** check box.

The results appear as follows:

<input checked="" type="checkbox"/> Unique Level Select a level in the hierarchy control to see the query items.			
Name	Role		Source
Product Name	_memberCaption	...	Products.Product Name
Product Number	_businessKey	...	Products.Codes.Product Number
Product Description	_memberDescription	...	Products.Product Description
Product Image		...	Products.Product Image
Introduction Date		...	Products.Introduction Date
Discontinued Date		...	Products.Discontinued Date
Production Cost		...	Products.Production Cost
Gross Margin		...	Products.Gross Margin

With this data, product names repeat in certain instances because there may be more than one color associated with the product. Sunglasses for instance come in several colors. In order to make the data understandable, you will concatenate the product number with the product name in the member caption.

8. In the **Product Name** row, click the **ellipses** in the **Source** column.
9. At the end of the expression, type `|| ' - ' || cast(`, from the **Available Components** pane, add **Product Number** to the end of the expression, and then type `, VARCHAR(6))`.

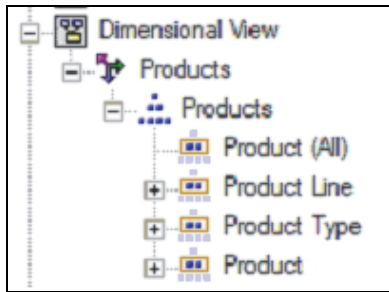
The results appear as follows:

```
[Consolidation View].[Products].[Product Name] || ' - ' || cast([Dimensional View].[New Dimension].[Products].[Product].[Product Number], VARCHAR(6))
```

10. Click **OK**, and then rename **Product Name** to **Product Caption**.

11. Click **OK**, rename the new dimension to **Products**, and then save the project.

The results appear as follows:



Task 3. Create a regular dimension from the time dimension.

1. Create a **Time** regular dimension as follows:
 - Create a new **Regular Dimension** in **Dimensional View**.
 - From **Consolidation View>Time**, drag **Year**, **Quarter Key**, **Month Key**, and **Day Key** into the **Hierarchies** pane.
 - Rename the levels as follows: **Time**, **Time (All)**, **Year**, **Quarter**, **Month**, **Day**.
 - For **Quarter**, **Month**, and **Day** levels, select the **Unique Level** check box.
 - For **Year** through **Day** levels, set or ensure the **Role** is **_businessKey**.
 - For **Month**, drag **Month (numeric)** to the bottom right pane with no role.
 - Click **OK** and then rename the regular dimension to **Time**.
2. Right-click **Time**, click **Verify Selected Objects**, and then click **Verify Model**.
 Four error messages appear, indicating that the levels do not have captions specified. You will set Year in the Year level as both business key and caption.

3. Click **Close**, and then double-click **Time** to re-open the **Dimension Definition** dialog box.
4. In the **Hierarchies** pane, click **Year**, and then, in the bottom pane, click the **ellipsis (...)** in the **Role** column.
5. Select **_memberCaption**, click **Close**, and then click **OK**.
6. Right-click **Time**, click **Verify Selected Objects**, and then **Verify Model**.
There are still four errors. The error related to the Year level, however, has changed. It indicates that you cannot assign the **_memberCaption** role to a data type that is not "string". You will fix this in the next task.
7. Click **Close**, and then save the project.

Task 4. Create string calculations for member captions.

1. In the **Dimensional View** namespace, double-click **Time** to re-open the **Dimension Definition** dialog box.
2. In the **Hierarchies** pane, click **Year**, and then clear the **_memberCaption** check box in the **Role** dialog box.
3. Click **Close**, and with **Year** still selected, click **Add** in the bottom right corner.
4. In the **Name** box, type **Year Caption**, and then, in the **Expression definition** pane, type **cast(**.
5. In the Available Components pane, expand **Consolidation View>Time**.
6. Double-click **Year** to add it to the **Expression definition** pane, and then type the following: **, VARCHAR(4))**.

The results appear as follows:

```
cast([Consolidation View].[Time].[Year] , VARCHAR(4))
```

7. Click **Test Sample** to verify the results, click **OK**, and then set the **Role** for **Year Caption** to **_memberCaption**.

8. Repeat steps **3** to **7** to create the following calculations to act as the member caption for their respective levels with the appropriate expression:

Calculation Name	Expression
Quarter Caption	<code>cast([Consolidation View].[Time].[Year], VARCHAR(4)) ' Q' cast([Consolidation View].[Time].[Quarter], VARCHAR (2))</code>
Month Caption	<code>[Consolidation View].[Time].[Month]</code>
Day Caption	<code>cast([Consolidation View].[Time].[Date], VARCHAR (10))</code>

9. Click **OK**, and then verify the selected object.
There are no errors.
10. Save the project, and leave Framework Manager open for the following workshop.

Results:

By creating regular dimensions, you started to develop a DMR model that will be used in a new business view to meet OLAP-style querying requirements.

Workshop 1: Create a Regular Dimension

You have been asked to add another dimension for the new Dimensional View. Authors must be able to analyze staff as well as products and time against sales and sales target measures.

To accomplish this:

- Create a regular dimension called Staff by Location based on the Staff by Location query subject in the Consolidation View.
- Create a hierarchy called Staff by Location with the following levels and attributes:
 - Staff by Location (All)
 - Staff Region
 - Staff Region Code (business key), Staff Region (member caption, renamed to Staff Region Caption)
 - Staff Country (unique level)
 - Staff Country Code (business key), Staff Country (member caption, renamed to Staff Country Caption)
 - Staff City (unique level)
 - Staff Branch Code (business key), Staff City (member caption, renamed to Staff City Caption), Staff Address 1 (no role), Staff Address 2 (no role), Staff Prov/State (no role), Staff Postal Zone (no role)

STAFF NAME (UNIQUE LEVEL)

- Sales Staff Code (business key), Staff Full Name (member caption, renamed to Staff Name Caption), First Name (no role), Last Name (no role), Work Phone (no role), Extension (no role), Fax (no role), Email (no role), Manager (no role), Position (no role)

For more detailed information outlined as tasks, see the Task Table on the next page.

For the final query results, see the Workshop Results section that follows the Task Table.

Workshop 1: Task Table

Task	Where to Work	Hints
1. Create a regular dimension named Staff by Location.	Project Viewer pane, Dimension View, Dimension Definition dialog box	<ul style="list-style-type: none"> • Create a new regular dimension in Dimensional View. • Add the following levels from Consolidation View>Staff by Location: Staff Region, Staff Country, Staff City, Staff Full Name • Rename the hierarchy and the top level to Staff by Location and Staff by Location (All). • Rename the Staff Full Name level to Staff Name. • Set the Staff Country, Staff City, and Staff Name levels as unique.

<p>2. Fine-tune the new dimension.</p>	<p>Project Viewer pane, Dimension View, Dimension Definition dialog box</p>	<ul style="list-style-type: none"> • Staff Region level: <ul style="list-style-type: none"> • ensure Staff Region is the member caption, and then rename it to Staff Region Caption to maintain your naming convention • add Staff Region Code as the business key • Staff Country level: <ul style="list-style-type: none"> • ensure Staff Country is the member caption, and then rename it to Staff Country Caption • add Staff Country Code as the business key
--	---	---

2. (Cont'd)		<ul style="list-style-type: none"> • Staff City level: <ul style="list-style-type: none"> • ensure Staff City is the member caption, and then rename it to Staff City Caption • add Staff Branch Code as the business key • add Staff Address 1, Staff Address 2, Staff Prov/State, Staff Postal Zone, (no role for each) • Staff Name level: <ul style="list-style-type: none"> • ensure Staff Full Name is the member caption, and then rename it to Staff Name Caption • add Sales Staff Code as the business key • add First Name, Last Name, Work Phone, Extension, Fax, Email, Position, Manager (no role for each) • Rename the new dimension to Staff by Location, organize alphabetically, and then save the project.
-------------	--	---

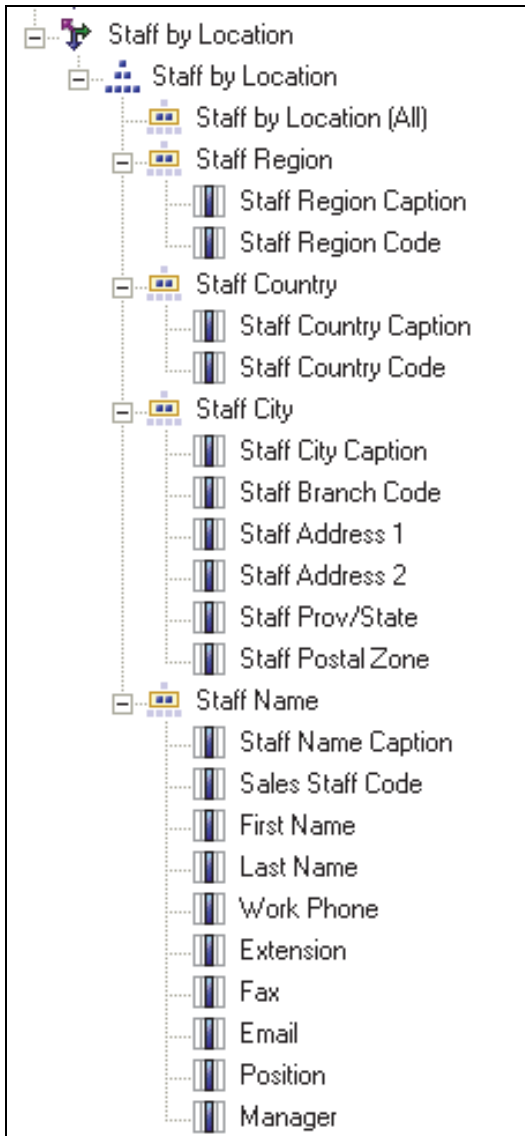
Workshop 1: Workshop Results

Your Staff by Location hierarchy appears as shown below:

Hierarchies:
Staff by Location
Staff by Location (All)
Staff Region
Staff Country
Staff City
Staff Name

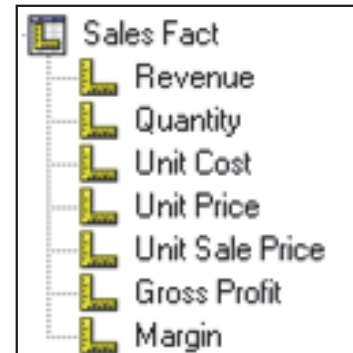
Workshop 1: Workshop Results

Your Regular Dimension, when finished and expanded, appears in the Project Viewer as shown below:



Define Measure Dimensions

- A logical collection of facts which enables OLAP-style analytical querying
- Related to regular dimensions within scope
- Can be created from:
 - a single table in a database
 - multiple tables across multiple databases

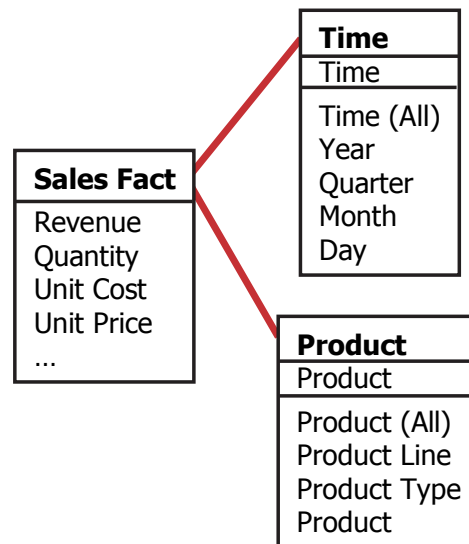
Cognos.
software

© 2010 IBM Corporation

Measure Dimensions are related to Regular Dimensions through scope relationships, that define at what levels a measure is in scope. However, underlying join relationships are required to generate the SQL that is sent to the data source.

Define Scope Relationships

- Exist between measure dimensions and regular dimensions
- Define the dimensions, hierarchies and levels in scope for measures
- Can be created, modified or deleted



A scope relationship is automatically created between a dimension and a measure dimension whose underlying query subjects have a valid JOIN relationship defined.

Scope relationships are required between measures and their related dimensions to achieve predictable rollups.

Scope relationships are not the same as join relationships. They do not impact the WHERE clause of the generated SQL, but rather which levels in a dimension are available for reporting for a particular measure.

Scope relationships define which regular dimensions are included by default in star schema groupings.

Edit DMR in the Dimension Map

- View, create, or modify:
 - regular or measure dimensions
 - hierarchies or levels
 - scope relationships

Dimensions - Scope Mode (Multiple)		
Products	Staff by Location	Time
Products	Staff by Location	Time
Product (All)	Staff by Location (All)	Time (All)
Product Line	Staff Region	Year
Product Type	Staff Country	Quarter
Products	Staff City	Month
	Staff Name	Day

© 2010 IBM Corporation

Cognos.
software

The Dimension Map view displays all regular and measure dimensions contained in a namespace and can be directly modified in this view.

If a measure is not in scope for a particular level of a regular dimension, you will see blank values in Analysis Studio, or repeating values (based on the values found at the parent level) in Query Studio or Report Viewer, but the values are not double counted.

You can edit scope relationships for either measure dimensions or individual measures within a measure dimension (in cases where the fact table has multiple levels of granularity).

Demo 2: Create Measure Dimensions, Set Scope, and Create a Presentation View

Purpose:

To continue the development of the DMR view required by authors and business analysts, you will create two measure dimensions, one for sales measures and one for sales target measures.

You will need to specify the scope of the measures and then populate a new Presentation view before testing the DMR portion of the model.

Components: Framework Manager, Analysis Studio

Project: GO Operational

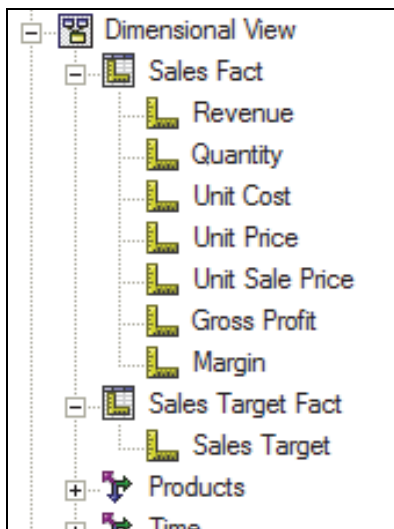
Package: GO Operational (analysis)

Task 1. Create Sales Fact and Sales Target Fact measure dimensions.

1. Right-click **Dimensional View**, point to **Create**, and then click **Measure Dimension**.
2. In the **Model Objects** pane, expand **Consolidation View** and **Sales Fact**.
3. Click **Revenue**, Shift+click **Margin**, and then drag all the selected measures to the **Measures** pane.
4. Click **OK**, rename the new measure dimension to **Sales Fact**, and then move it above **Products**.
5. Repeat steps 1 to 4 to create a measure dimension called **Sales Target Fact** based on the **Sales Target Fact** data source query subject in **Consolidation View**. Select only the **Sales Target** measure and move the new measure dimension below the **Sales Fact** measure dimension.

- Expand **Sales Fact** and **Sales Target Fact** to view the items.

The results appear as follows:



- Save the project.

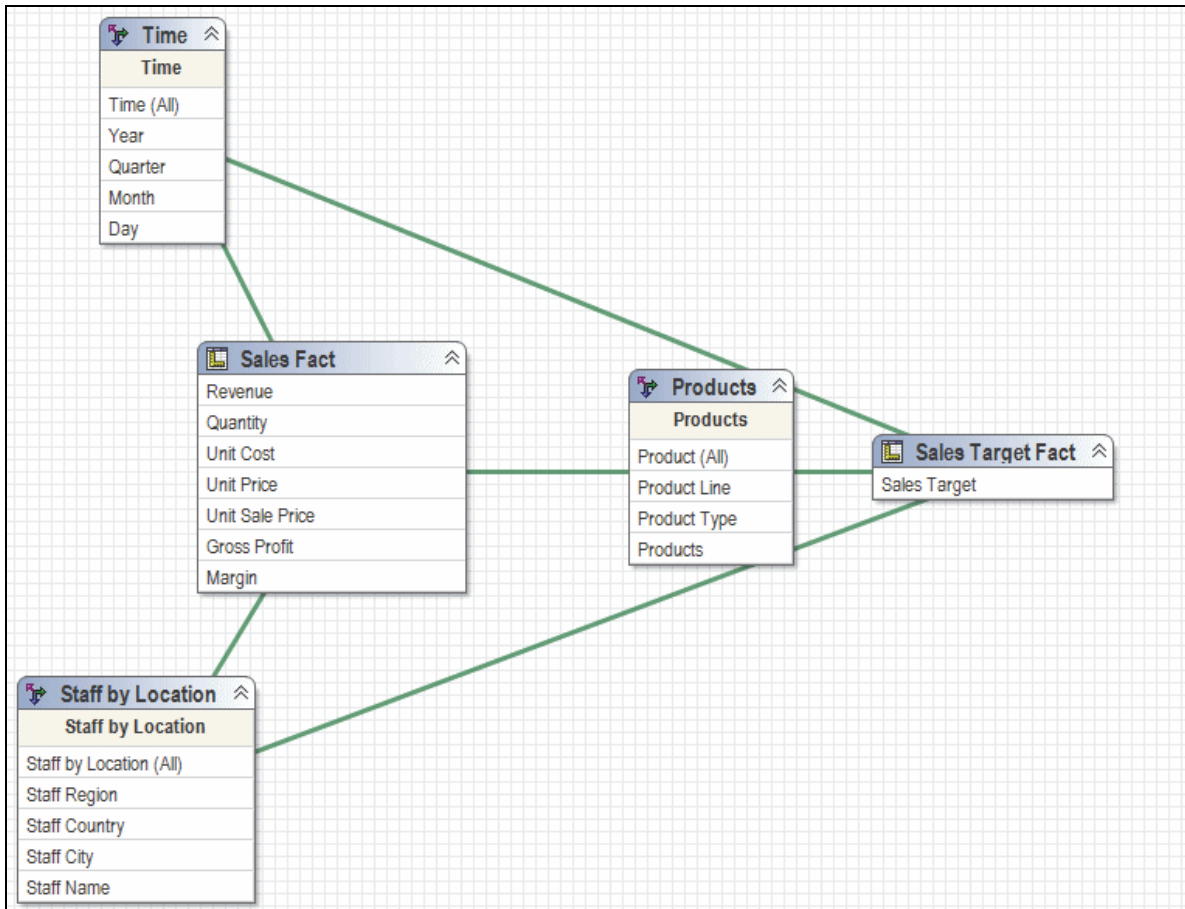
Task 2. Set scope for the measure dimensions.

First you will view the scope relationships in the Diagram pane.

- Double-click the **Dimensional View** namespace to give it focus, and then in the middle pane, click **Diagram**.
- From the **Diagram** menu, click **Diagram settings**, select **Scope Relationships**, and then click **OK**.

- On the **toolbar**, click **Auto Layout**, beside **Layout Style**, select **Star**, click **Apply** and then click **Close**.

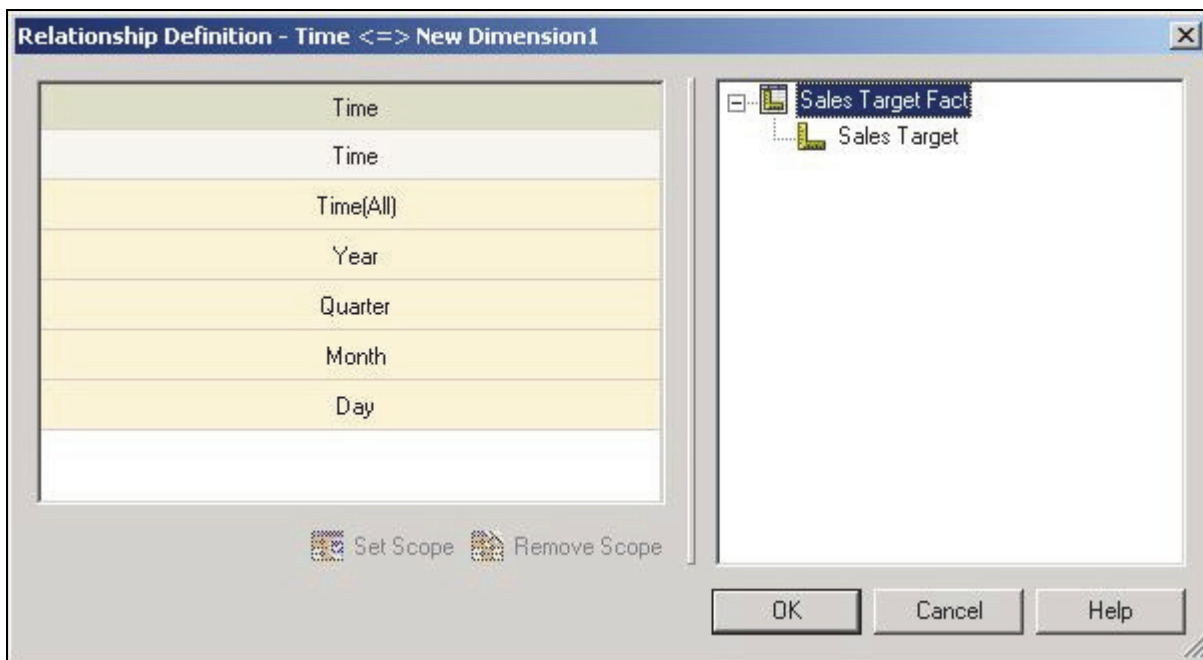
The results appear as follows:



The facts are joined to their related dimensions through scope relationships. These relationships were automatically generated based on the underlying relationships in the Foundation Objects View.

4. Double-click the scope relationship between **Time** and **Sales Target Fact** and then, in the right pane, click **Sales Target Fact**.

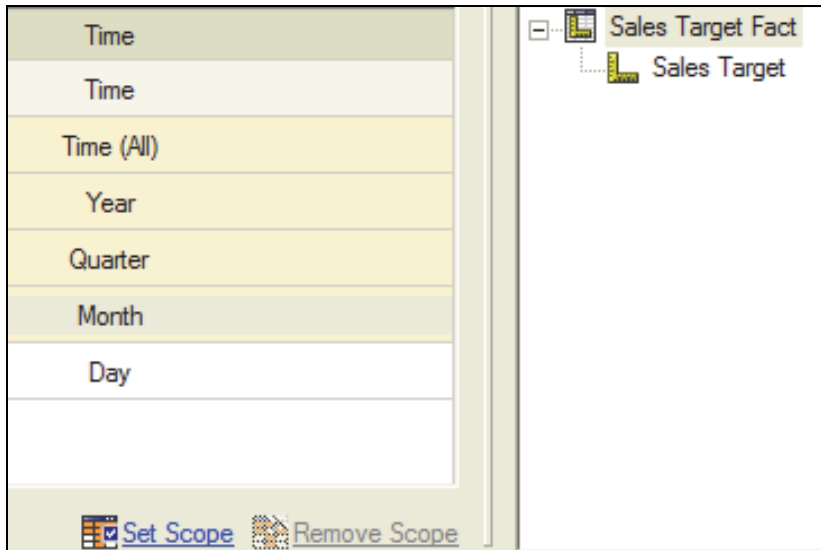
The results appear as follows:



All levels in the Time dimension are highlighted indicating that they are all currently in scope. This is not the case for sales targets since they are at the month level. You can set the scope in this dialog or in the Dimension Map pane. You will quickly set scope for this measure here but then cancel the changes and then set scope in the Dimension Map in order to learn both methods.

5. Click the **Month** level, and then click **Set Scope**.

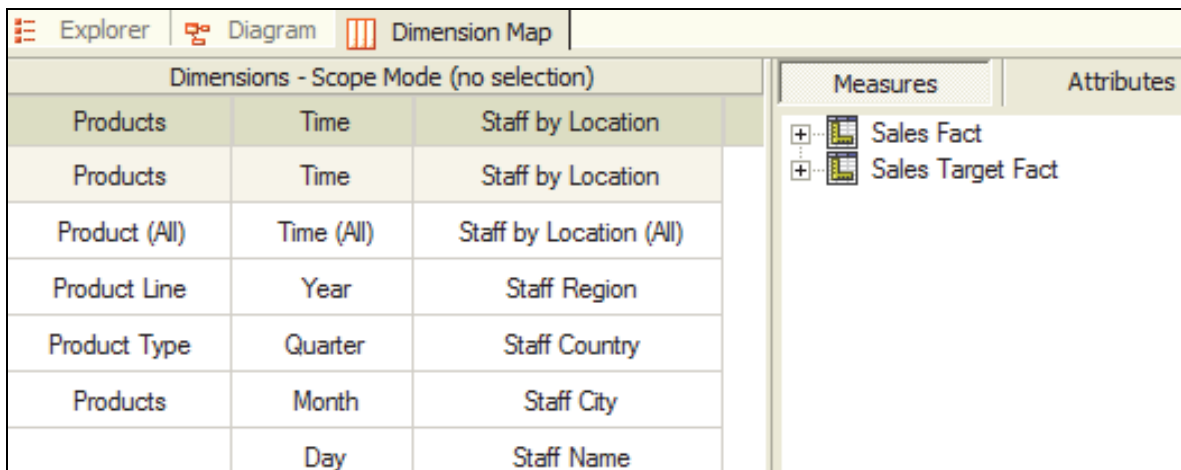
The results appear as follows:



The Day level is no longer highlighted and is now out of scope for the Sales Target measure.

6. Click **Cancel**, and then in the middle pane, click the **Dimension Map** tab.

The results appear as follows:



The three dimensions appear in the left pane and the measure dimensions appear in the right pane. Here you can create, edit, and delete dimensions and set scope.

7. In the **Measures** pane, click **Sales Fact**.

The results appear as follows:

Explorer Diagram Dimension Map			
Dimensions - Scope Mode (Multiple)			Measures Attributes
Products	Time	Staff by Location	<div> <div>+</div> <div>+</div> </div> Sales Fact Sales Target Fact
Products	Time	Staff by Location	
Product (All)	Time (All)	Staff by Location (All)	
Product Line	Year	Staff Region	
Product Type	Quarter	Staff Country	
Products	Month	Staff City	
	Day	Staff Name	

All regular dimensions are highlighted, indicating that all measures in Sales are in scope.

8. Click **Sales Target**.

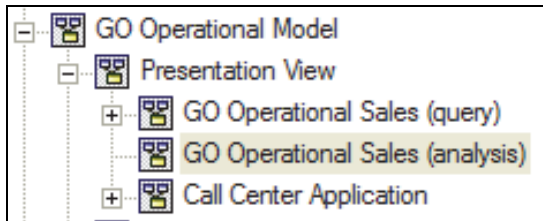
Again, the measure is in scope for all dimensions. You will set the scope for Sales Target Fact to be at the Month level for Time and Product Type level for Products.

9. Click **Month** in the **Time** dimension, and then on the toolbar, click **Set Scope** .

Task 3. Create a DMR Presentation view using star schema groupings.

1. In the **Presentation View**, create a namespace called **GO Operational Sales (analysis)**, and then drag below **GO Operational Sales (query)**.

The results appear as follows:



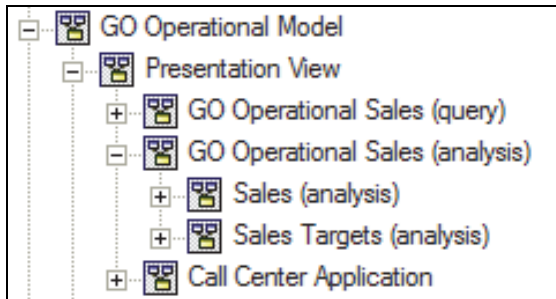
2. In the **Dimensional View**, right-click **Sales Fact**, and then click **Create Star Schema Grouping**.

Unlike when you created star schema groupings for relational metadata in the Consolidation View, here you do not need to select all the desired dimensions along with the fact. This is because Framework Manager uses the scope relationships to identify related dimensions. The Consolidation View objects have no relationships and therefore you needed to select all required objects. Had there been relationships, the Star Schema Grouping wizard would use them to detect related objects for the grouping.

3. Change the namespace name to **Sales (analysis)**, and then click **OK**.
4. Drag **Sales (analysis)** to **GO Operational Sales (analysis)**.

- Repeat steps 2 to 4 with **Sales Target Fact** to create **Sales Targets (analysis)**.

The results appear as follows:

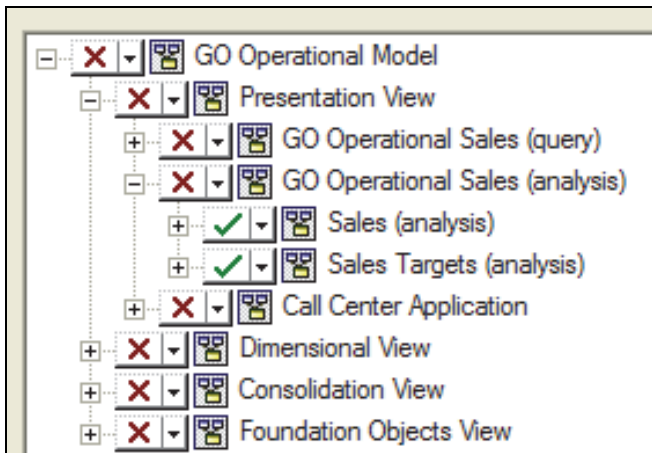


- Save the project.

Task 4. Create a GO Operational (analysis) package.

- Right-click **Packages**, point to **Create**, and then click **Package**.
- In the **Name** box, type **GO Operational (analysis)**, and then click **Next**.
- Clear the **GO Operational Model** check box, and then expand **Presentation View>GO Operational Sales (analysis)**.
- Select **Sales (analysis)** and **Sales Target (analysis)**.

The results appear as follows:



- Click **Finish**, click **Yes** to open the Publish Package wizard, clear the **Enable model versioning** check box, click **Next** twice, click **Publish**, and then click **Finish**.

The Verify Model dialog box appears listing informational messages.

- Click **Close**, and then save the project.

Task 5. Test the new package in Analysis Studio.

- Launch **IBM Cognos Connection**, log in, and then launch **Analysis Studio** selecting the **GO Operational (analysis)** package.
- Click **Blank Analysis**, and then click **OK**.
- In the **Insertable Objects** pane, expand **Sales (analysis)**, and then drag **Products** to the **Columns** drop zone in the analysis work area.
- Drag **Time** to the **Rows** drop zone in the analysis work area.
- Expand the **Sales Fact** measure dimension, and then drag **Revenue** to the **Measure** drop zone in the analysis work area.

The results appear as follows:

Revenue	Camping Equipment	Mountaineering Equipment	Personal Accessor
2004	\$332,986,338.06		\$39
2005	\$402,757,573.17	\$107,099,659.94	\$45
2006	\$500,382,422.83	\$161,039,823.26	\$59
2007	\$352,910,329.97	\$141,520,649.70	\$44
Time (All)	\$1,589,036,664.03	\$409,660,132.90	\$1,885

- Click the intersection of **Camping Equipment** and **2007** once to give **\$352,910,329.97** focus, and then click it again to drill down on both the row and the column at the same time.

7. Drag **Staff by Location** below **Cooking Gear** to nest location under **Products**.

The results appear as follows:

Rows:		Columns:			
2007		Camping Equipme...		Staff by Locati...	
Revenue	Cooking Gear				
	Americas	Asia Pacific	Northern Europe	Central Europe	Southern Europe
200701	\$7,267,580.19	\$6,255,436.62	\$3,154,394.07	\$5,624,638.61	\$2,760,20
200702	\$7,163,637.99	\$6,130,640.17	\$3,068,634.70	\$5,529,673.21	\$2,706,22
200703	\$2,502,059.11	\$2,171,358.39	\$1,092,996.10	\$1,932,536.73	\$953,78
2007	\$16,933,277.29	\$14,557,435.18	\$7,316,024.87	\$13,086,848.55	\$6,420,21

You can now further analyze sales in relation to location.

8. Drill down on the intersection of **Cooking Gear**, **Central Europe** and **2007Q2**, and then on **June**.
9. In the Insertable Objects pane, expand **Sales Target (analysis)>Sales Target Fact**.
10. Drag **Sales Target** to the measures section of the analysis to replace the existing measure.

The results appear as follows:

Sales target	Cooking Gear		
	France	Germany	Switzerland
20070601			
20070602			
20070603			
20070604			

Notice that the measure values are blank. This is because the Sales Target measure is not in scope at the Day level.

11. Click **June** to drill up.

Now values appear because you are at the Month level, which is in scope.

12. Leave **Analysis Studio** open for the next demo.

Results:

You have successfully completed a dimensionally modeled relational model by incorporating measure dimensions and specifying scope.

Member Sorting

- You can configure the default sort order for DMR members:
 - metadata (in metadata trees)
 - data (in reports)
- Can sort for OLAP compatibility
 - for example, ensure the following function returns the correct value:

```
parallelPeriod([Sales (analysis)].[Time].[Time].[Year],1,[2007-01-09])
```

In a dimensionally modeled relational (DMR) source, by default, the members returned under a level are returned in an order dependent on the data source provider.

Framework Manager allows the modeler to specify sorting options for the levels of a hierarchy.

You can specify how members are sorted in applicable studio metadata trees as well as how the members are sorted in report results.

Various dimensional functions used in Report Studio require the members to be in a specific order to achieve predictable results. In these cases, you can also choose to sort the members for OLAP compatibility to ensure the members are always returned in the same order based on your sort criteria. For example, if you want to compare sales for a particular date with the sales of the same day in the previous year (`parallelPeriod` function), you need to ensure that your time dimension date values are in chronological order.

Demo 3: Metadata Tree and Member Sorting

Purpose:

In order to create a more intuitive and predictable environment for authors, you will use some of the new sorting features available in Framework Manager. Specifically, you will sort members by alphabetical in the metadata tree and in the reports as well as configure sorting to be compatible with OLAP style queries.

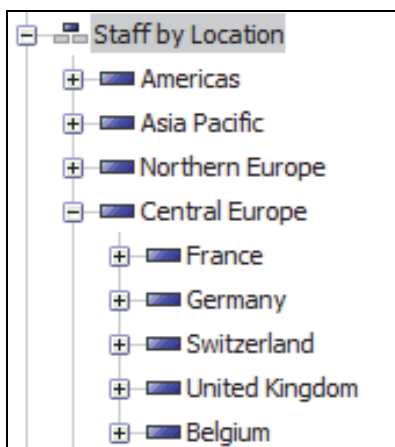
Component: Framework Manager, Analysis Studio, Report Studio

Package: GO Operational (analysis)

Task 1. Examine member sorting in Analysis Studio.

1. In **Analysis Studio**, in the data tree, expand **Staff by Location**>**Central Europe**.

The results appear as follows:



This data tree displays members. Notice that the members under Central Europe are not sorted alphabetically.

2. Expand **France**.

Again, the values below France are not sorted.

3. Examine the data in the analysis.

Sales Target	Cooking Gear				
	Germany	Switzerland	United Kingdom	Belgium	Central Europe
April	467,000	271,300	461,500	379,200	2,073,800
May	389,000	213,400	432,800	187,800	1,682,000
June	425,900	213,500	424,400	167,300	1,611,700
2007 Q2	1,281,900	698,200	1,318,700	734,300	5,367,500

Again, the members for Central Europe are not sorted alphabetically. You can change this default behavior in Framework Manager by configuring member sorting for the regular dimension.

4. Save the analysis in the **GO Operational (analysis)** folder as **Module 15 - Member Sorting Test**, and then close **Analysis Studio**.

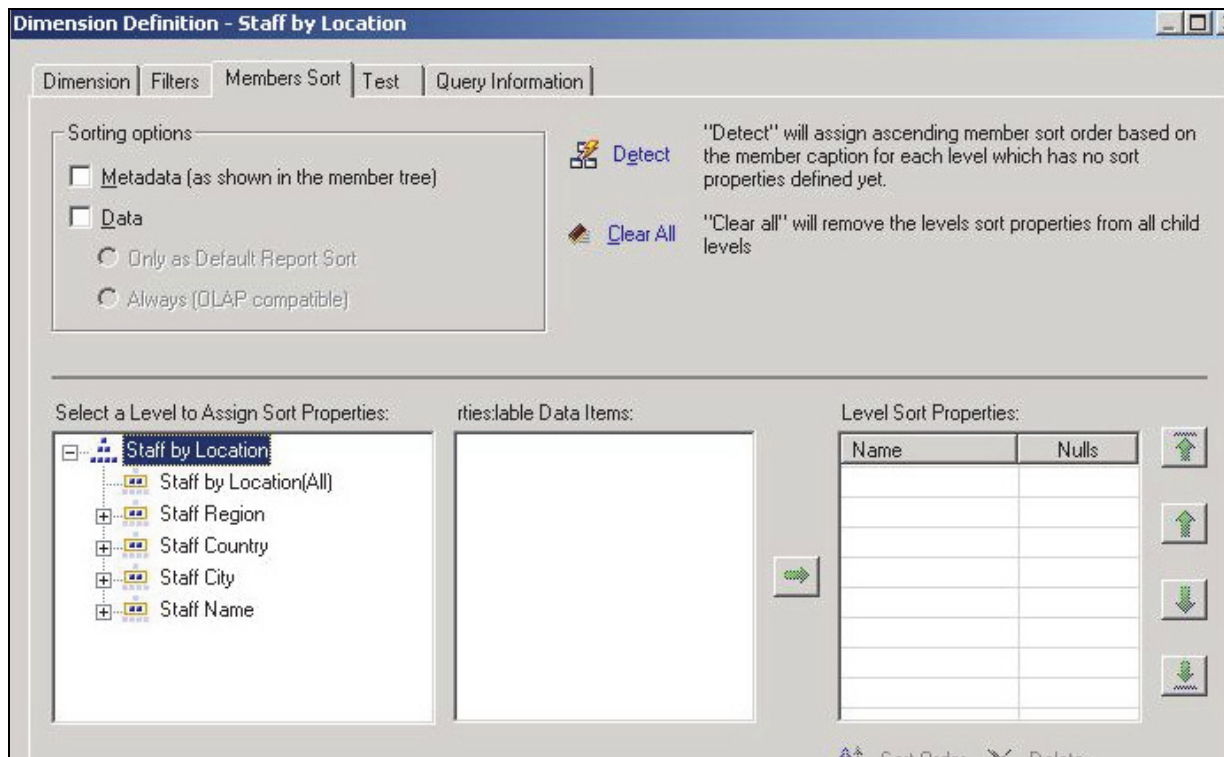
Task 2. Sort members in Framework Manager.

You will resolve two issues using the member sorting feature. One is to ensure that members appear sorted in the data trees in applicable studios and the other is to ensure that the time dimension is compatible with dimensional functions.

1. In **Framework Manager**, in the **Dimensional View** namespace, double-click **Staff by Location**.

2. Click the **Members Sort** tab.

The results appear as follows:



This tab is used to sort members in applicable metadata trees and in reports. The panes at the bottom of the dialog are used to configure the sort behavior. The options behave as follows:

Metadata: used to sort members of the level in the metadata tree.

Data - Only as Default Report Sort: used to sort members in the report according to the sort information specified on the levels.

Data - Always (OLAP compatible): used to provide member relative functions with a sorted structure of the members that can be navigated with consistency. The members of the level will also be sorted in the metadata tree and reports. You will now choose to sort the members of this regular dimension both in the data tree and in the reports.

- Under **Sorting Options**, select both the **Metadata** and **Data** check boxes, and then click **Detect**.

A message appears indicate member sort properties were added to four levels.

- Click **OK**, and then click **Staff Region**.

The results appear as follows:



Here you can see what criteria are used to sort the members for each level. In this case Staff Region Caption (the item used as the member caption) is used to sort the members. All the levels are currently using the member caption as the sort criteria. If you require the sorting to be applied on another item such as a key or some other attribute, you can edit the settings. You can also use multiple items in the sorting criteria. For example, you can sort first by name and then by code. If there are two identical names, then the code will be used to determine which name is displayed first.

You also have the option on where to display null values in your results. This is configured under the Nulls column. Selecting First places the null values at the beginning, and Last places the null values at the bottom. Unspecified uses the setting defined in the data source.

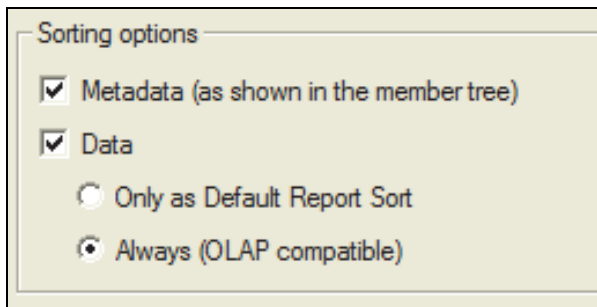
- Click **OK**.

You will now ensure that the Time dimension members are compatible with dimensional functions.

- Double-click **Time**, and then click the **Members Sort** tab.

- Under **Sorting Options**, select both the **Metadata** and **Data** check boxes, and then select **Always (OLAP compatible)**.

The results appear as follows:



Sorting options

☒ Metadata (as shown in the member tree)

☒ Data

☐ Only as Default Report Sort

☒ Always (OLAP compatible)

- Click **Detect**, click **OK**, and then click **OK** again.
- Save the project.

Task 3. Re-test member sorting in the studios.

- Publish the **GO Operational (analysis)** package.
- In **IBM Cognos Connection**, click **GO Operational (analysis)**, and then click **Module 15 - Member Sorting Test**.

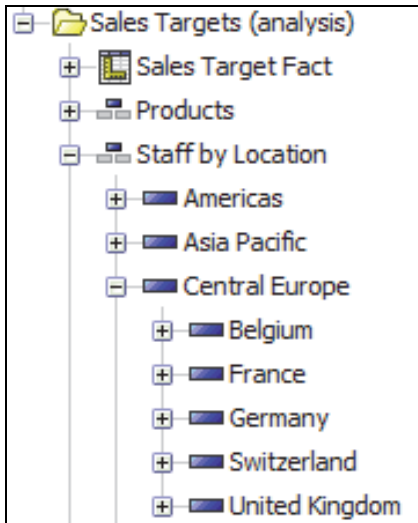
The results appear as follows:

Sales Target	Cooking Gear				
	Belgium	France	Germany	Switzerland	United Kingdom
April	379,200	494,800	467,000	271,300	461,500
May	187,800	459,000	389,000	213,400	432,800
June	167,300	380,600	425,900	213,500	424,400
2007Q2	734,300	1,334,400	1,281,900	698,200	1,318,700

Notice the members in the analysis for Central Europe are now sorted.

3. In the data tree, expand **Sales Targets (analysis)>Staff by Location>Central Europe**.

The results appear as follows:



The members are sorted here as well. If you continue to expand the members in the data tree for this dimension, you will see all levels are sorted.

Note: When looking at lower level members in the data tree in Report Studio, they will be sorted but there will be logical breaks where the parent level changes and the sorting will appear to begin again. For example, if you look at members sorted alphabetically at the Product Type Level, you will see members sorted alphabetically for Camping Equipment first, and then you will next see members sorted alphabetically for Golf Equipment. In a pseudo pattern it would look like (abcdefabcdef).

4. Close **Analysis Studio**.

- Launch Report Studio and create a crosstab report with the following data Items from the **Sales (analysis)** namespace.

Products Dimension > **Products** > **Product Line** to **Columns**

Time Dimension > **Time** > **Day** to **Rows**

Sales Fact Dimension > **Revenue** to **Measures**

- Run the report.

The results appear as follows:

Revenue	Camping Equipment	Personal Accessories	Outdoor Protection	Golf Equipment	Mountain Equipment
20040000					
20040101					
20040102					
20040103					
20040104					
20040105					
20040106					
20040107					
20040108					
20040109					
20040110					
20040111					
20040112	\$20,217,372.98	\$7,414,443.06	\$2,263,380.47	\$9,141,599.89	
20040113	\$5,000,710.60	\$3,477,197.59	\$474,025.75	\$2,536,524.65	
20040114	\$633,110.20	\$2,118,932.80	\$91,322.21	\$388,795.27	
20040115	\$737,487.22	\$1,858,835.02	\$62,434.09	\$421,685.11	
20040116	\$279,804.62	\$1,557,191.77	\$80,466.92	\$59,433.38	
20040117					

The dimensional function now returns the correct date from the previous year.

This is because you configured this regular dimension to sort members for OLAP compatibility. This may require more processing, but the results are predictable. If performance becomes an issue for large dimension tables using this technique, you may consider sorting at the table level.

7. Close the browser, and then, in **Framework Manager**, close and save the project.
8. Close Framework Manager.

Results:

By using the new member sorting features in Framework Manager, you were able to create a more intuitive and predictable experience in the studios for authors.

Note: This demo illustrated a relative time function (`parallelPeriod`). It is worth noting that although you can easily perform relative time analysis with dimensional functions in Report Studio, ad hoc query users in Query Studio cannot leverage this functionality. There are modeling techniques to allow for relative time analysis without the use of dimensional functions. Refer to Appendix C for one such technique.

Dimensional Modeling Considerations

- Dimensional objects provide an additional layer of metadata that enables OLAP behaviors
- It may be necessary to employ a form of database vendor materialization to improve performance
- Build mandatory filters into your model to ensure that end users do not accidentally retrieve excessively large data sets

The rules regarding data volumes that apply to building cubes also apply to a DMR source. The key difference is that with filtering strategies, you can perform analysis against larger volumes in a relational source than is practical to do with most OLAP sources.

Modeling Recommendations Review

1. Define reporting requirements and data access strategies
 2. Import only required reporting objects in a phased approach and alter as little as possible
 3. Verify relationships and query item properties
 4. Model in freehand to identify query usage
 5. Use model query subjects to control query generation and usage and to consolidate metadata
-
- cont'd...

At this point in the course you have touched on all of the modeling recommendations. This is a good time to review each of the recommendations in more detail.

1. Before beginning any modeling exercises, determine what the reporting requirements are. This will help you to find the correct data and define a data access strategy.

Based on available data sources, data volumes, and environmental factors such as network speed, hardware processing power, and so on, an appropriate data access strategy should be planned and implemented to ensure acceptable response times to report requests.

2. Import only what is required for reporting and import it in manageable chunks. Alter the data source query subjects as little as possible. Leaving the data source query subjects as simple all-inclusive select statements reduces future maintenance. For example, when a table has a new column added to it, simply update the data source query subject that references it in Framework Manager and the new column will appear as a new query item.
3. Verify that the relationships created during an import reflect those in the data source and that the query item properties are set correctly.

For relationships, notice:

- cases where a dimension query subject relates to a fact query subject on different keys
 - cases where there are multiple valid relationships between query subjects
 - dimension query subjects that belong to multiple hierarchies
4. Model in freehand to identify modeling challenges and how query subjects are used (which query subjects are treated as facts, dimensions, or both). Identifying these issues on paper can provide a clear modeling plan.
 5. Begin creating simplified, abstracted model query subjects to resolve modeling challenges by:
 - creating aliases where required to control query paths
 - modeling as a virtual star schema to control SQL generation (what is a fact, what is a dimension)
 - removing descriptive (dimensional) attributes from fact tables
 - consolidating related information into one model query subject for a cleaner presentation (for example, placing all product related query items in one model query subject)

Modeling Recommendations Review (cont'd)

6. Customize metadata for run time
7. Specify determinants as required
8. Resolve multiple ambiguous joins between query subjects
9. Create analysis objects if OLAP-style querying is required
10. Create the business view as a set of star schema groupings

6. Customize metadata for runtime by using:
 - parameter maps and session parameters to handle dynamic column or row retrieval
 - prompt values and query macros to add mandatory user prompts and security filters
7. Specify determinant information where required to enable accurate aggregation in cases where a level of granularity has repeating keys, your data contains BLOBs, or you wish to avoid the distinct clause on unique values when grouping or enhance performance for regular dimensions.
8. Resolve any relationship ambiguities, such as multiple joins between two query subjects, by deleting surplus joins and by creating role-playing dimensions.
9. Create regular and measure dimensions if authors need to perform OLAP-style queries on relational data.
10. Use star schema groupings to build logical business groupings in the business view and to indicate conformed dimensions based on naming conventions.

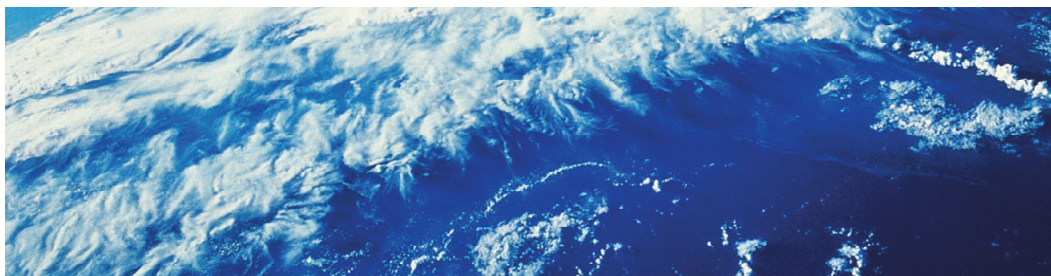
Summary

- You should now be able to:
 - apply dimensional information to relational metadata to enable OLAP-style queries
 - sort members for presentation and predictability



Manage MUNs in Framework Manager

IBM Cognos BI



Business Analytics

© 2010 IBM Corporation

Objectives

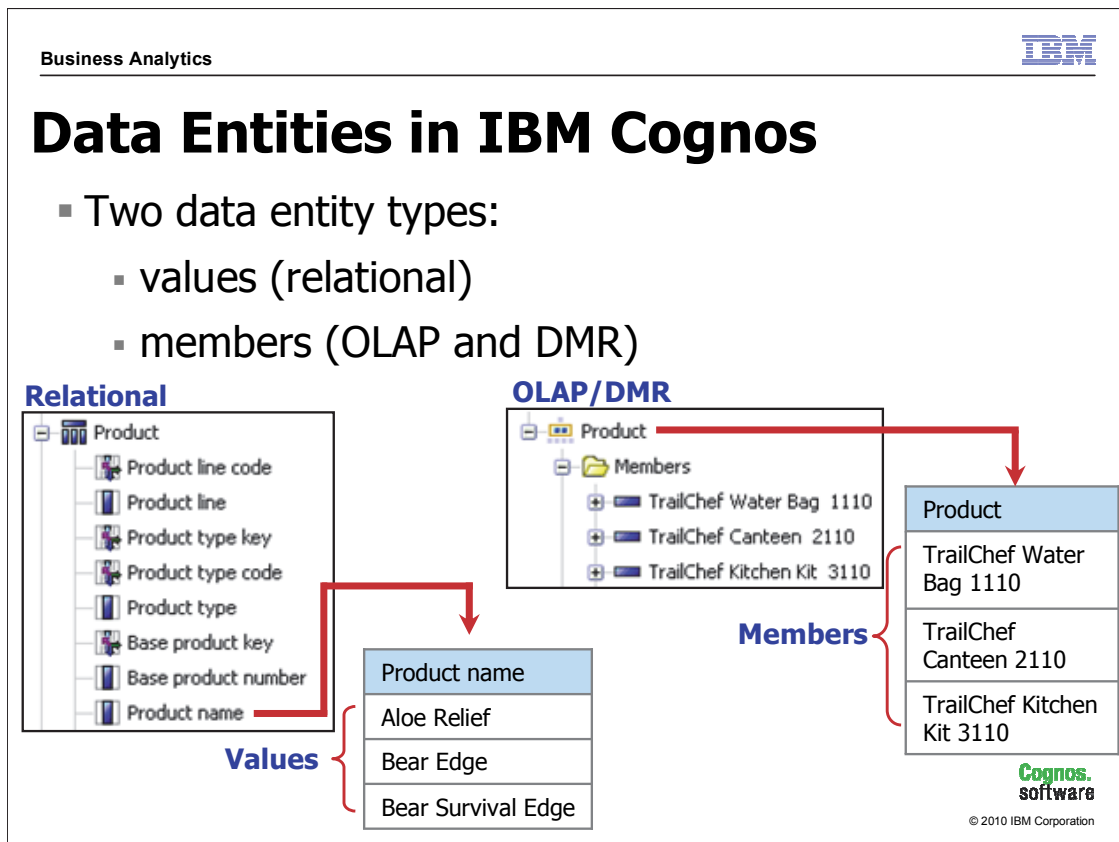
- At the end of this module, you should be able to:
 - describe model types and data entities
 - define members and member unique names
 - identify changes that impact a MUN

Model Types in IBM Cognos

- IBM Cognos lets you work with three model types:
 - relational
 - dimensionally modeled relational (DMR)
 - OLAP

Relational models have a basic metadata structure that looks like tables and columns in a database.


DMR and OLAP models display metadata in a multidimensional structure that is comprised of dimensions, hierarchies, and measures.



When an author creates a report with a relational model, the data values are retrieved from rows and columns from a table in the data source.

Members are entities from a multidimensional data structure. Each member has certain properties such as a member key and member caption. The values presented to authors and consumers, by default, are the member caption.

Members also come with additional context that describes their position in the multidimensional structure. This context is presented in the member unique name (MUN) of each member.

Business Analytics


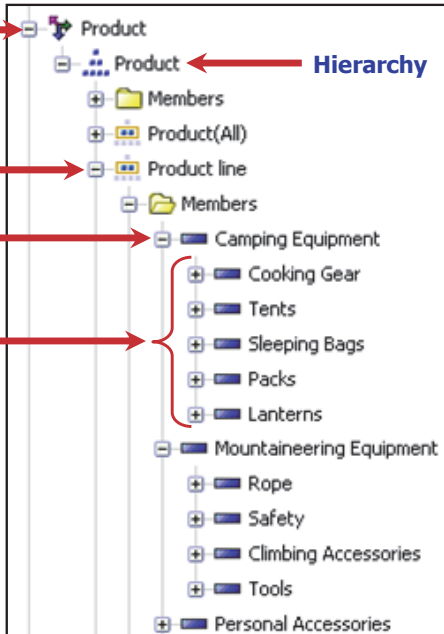
Work with Members

Dimension →

Level →

Member →


Child members →



- Members are located in levels of an OLAP or DMR structure.

Members used as data items for a report

Cooking Gear	Rope	Safety
<Cooking Gear>	<Rope>	<Safety>
<Cooking Gear>	<Rope>	<Safety>
<Cooking Gear>	<Rope>	<Safety>



© 2010 IBM Corporation

A dimension contains one or more hierarchies. Each hierarchy contains one or more levels. Each level contains one or more members. Each member can have child members, which are also found as members of the next lower level in the hierarchy.

Report Studio and Analysis Studio have members in their data trees and let authors work directly with the members. These studios are "member-aware". Query Studio is not "member-aware" and therefore does not have members in its data tree.

All studios let authors create reports using levels, which return all members of that level. If the studio in which you create reports is "member-aware", members can be used independently as data items.

The metadata items (member attributes) from the multidimensional model can also be used for report creation.

What Makes Up a Member?

- Framework Manager DMR source
 - member key = `_businessKey` (query item role)
 - member caption = `_memberCaption` (query item role)
- OLAP source (PowerCube)
 - member key = Category code
 - member caption = Category label
- OLAP source (DB2)
 - member key = Member Key
 - member caption = Member Name

The member key is used to identify a particular member in a multidimensional structure and can be used as a value in drill through and master-detail operations.

The member caption is the name that is displayed for the member.

Members may also have attributes such as alternate member names or other descriptive information.

What is a Member Unique Name (MUN)?

- Used to locate a member
- Referenced in an expression when a member is used:
 - in a report
 - in a filter or calculation
 - for drill-through

Framework Manager modelers, when modeling for DMR, do not create a MUN for a member. They specify what will be used as the member key and member caption. The member key will be used in the MUN when the MUN is generated at run time.

MUNs ensure that members are unique within the multidimensional structure.

What Makes Up a MUN?

- MUN from a PowerCube data source:
 - [great_outdoor_sales_en].[Products].[Products].[Product type]->:[PC].[@MEMBER].[101]
- MUN from a Framework Manager DMR data source:
 - [Sales].[Product].[Product].[Product type]
->[all].[2101].[101]

The MUN for the PowerCube data source in the slide example can be identified as follows:

- Level unique name:

↑

cube

↑

dimension

↑

hierarchy

↑

level

[great_outdoor_sales_en].[Products].[Products].[Product type]
- Type of cube: ->:[PC] This represents a PowerCube data source.
- Vendor specific MUN: [@MEMBER].[101]. Used by the data source to locate the requested member.

Changes that Impact a MUN

- MUNs can change when:
 - hierarchies or levels change
 - member keys change
 - category codes in PowerCubes
 - Member Key Column in MSAS cubes
 - _memberKey role in DMR models
 - members no longer exist in the data source
 - a production environment has more members than in the test environment

When MUNs change, they impact the reports that directly reference the members to which they point. Those MUNs must be identified and fixed in the report.

For drill-through scenarios, once a broken MUN reference is fixed, there is potential for the report to pass the wrong parameter to the target report. This can occur when the member key changes. This is why it is not recommended to change member keys. It is critical that business keys are conformed across the business ensuring that they do not change and that there is no need to change them.

If you understand what changes MUNs, you can model the metadata so that reports created in the test/development environment will run without problems in the production environment.

If a report references a level, which returns all members, then a changed MUN will not affect the report. Levels are not members and therefore do not have a MUN.

It is not recommended for authors to try and manually construct MUNs.

Demo 1: Identify How Changes to MUNs Impact Reports

Purpose:

Before you move to our production environment, you will use our test environment to identify how changes to a model impact reports that use members. You will begin by accessing a DMR source to create a report. You will create the report using members as data items, and then identify a MUN for a member in the report. You will then make a change to the model that will impact the MUN, re-publish the package, and then re-run the report. You will identify how the report is impacted, and subsequently how the MUN is impacted. You will then fix the report.

Components: Framework Manager, Report Studio

Project: great_outdoors_warehouse

Package: GO Data Warehouse (analysis)

For this demo you will use a new model called great_outdoors_warehouse based on a reporting database (star schema) for the Great Outdoors Company.

Task 1. Publish a model.


1. In **Framework Manager**, close any projects that may be open, and then open the great_outdoors_warehouse project located at **C:\Edcognos\B5152\CBIFM-Start Files\Module 16\great_outdoors_warehouse**.
2. If prompted, log in as User ID **admin**, and Password **Education1!**.
3. Publish the **GO Data Warehouse (analysis)** package, and ensure that the **Enable model versioning** is cleared.

Task 2. Create a report using members.

1. Launch **IBM Cognos Connection**, log on, and then launch **Report Studio** selecting the **GO Data Warehouse (analysis)** package.
2. Click **Create New**, and then double-click **Crosstab**.
3. In the **Insertable Objects** pane, expand **Sales and Marketing (analysis)>Sales target>Sales target fact**, and then drag the **Sales target** measure to the **Measures** drop zone in the report.
4. Expand **Employee (by position)>Employee (by position-department)>Position-department (level 1)>Members>Executive>Operations>Sales>Level 3 Sales Representative**, click the first member, **Shift+click** the third member, and then drag the selected items to the **Rows** drop zone.
5. Drag the **Employee (by position-department)(All)** level, to the **Rows** drop zone under the existing rows.
6. Expand **Time dimension>Time dimension**, and then drag the **Year** level to the **Columns** drop zone.

The results appear as follows:

Sales target	<#Year#>	<#Year#>
<#Aiko Watanabe#>	<#1234#>	<#1234#>
<#Akemi Yamada#>	<#1234#>	<#1234#>
<#Alessandra Torta#>	<#1234#>	<#1234#>
<#Employee (by position-department)(All)#>	<#1234#>	<#1234#>

7. On the toolbar, click **Run Report** .

The results appear as follows:

Sales target	2004	2005	2006	2007
Aiko Watanabe	12,260,000	9,870,300	12,648,200	4,575,700
Akemi Yamada	5,309,700	8,681,600	12,219,000	8,031,660
Alessandra Torta	7,408,000	7,996,500	8,136,100	7,529,800
Employee (by position-department)(All)	812,885,300	1,036,923,300	1,332,553,100	1,023,006,840

The report contains the values of the member items that you added during design.

8. Close **Cognos Viewer**.
9. In the **Insertable Objects** pane, right-click the **Aiko Watanabe** member, and then click **Properties**.

Notice the Member Unique Name property:

[Sales target].[Employee (by position)].[Employee (by position-department)].[Employee]->[all].[100].[220].[390].[43639].[4116]

At the very end of the MUN, the `_businessKey` role value used is 4116. This value is based on the Employee key in the data source.

10. Click **Close**.
11. Save the report as **Module 16 - MUN Test in GO Data Warehouse (analysis)**.
12. Close **Report Studio**.

Task 3. Change the **_businessKey** role for a level within a dimension.

1. In **Framework Manager**, in the **Project Viewer**, expand **go_data_warehouse>Dimensional view**.
2. Double-click **Employee (by position-department)**.
3. In the **Hierarchies** pane, click the **Employee** level.
In the bottom pane, the **_businessKey** role is set on **Employee** key. The values represented by this item are those that appear as the member key for the **MUN** of a member as shown earlier.
4. Under the **Role** column, click the **ellipsis** beside **Employee code**.
5. In the **Specify Roles** dialog box, select the **_businessKey** check box, and then click **OK** to the warning message.
6. Click **Close**, and then click **OK**.
7. Save the project in **C:\Edcognos\B5152\Course_Project\Great Outdoors Warehouse**, and then re-publish the **GO Data Warehouse (analysis)** package, overwriting the existing package.


Task 4. Examine the impact of a modeling change on the MUN for a member.

1. In **IBM Cognos Connection**, on the root of **Public Folders**, click the **GO Data Warehouse (analysis)** folder, and then click the **Module 16 - MUN Test** report.

The results appear as follows:

Sales target	2004	2005	2006	2007
Employee (by position-department)(All)	812,885,300	1,036,923,300	1,332,553,100	1,023,006,840

Data appears to be missing. The report does not contain the values of the member items that you added during design. It contains only the values for the metadata item you added (Employee(by position-department)(All)). The report is running against the most recent version of the package.

2. At the top of the page, click **Open with Report Studio** , click **OK**, and then click **OK** again.
3. In the **Insertable Objects** pane, expand **Sales and Marketing (analysis)>Sales target>Employee (by position)>Employee (by position-department)>Position-department (level 1)>Members>Executive>Operations>Sales>Level 3 Sales Representative**.
4. Right-click the **Aiko Wantanabe** member, and then click **Properties**.

Notice the Member Unique Name property. At the very end of the MUN, the `_businessKey` role value used now is 10572. This value is based on the Employee code in the data source. This MUN is now different based on the change you made to the model. The current employee members in the report layout are associated with MUNs that no longer exist and therefore are not returned in the report.

To correct this you must replace the existing members in the report layout with the current members from the Insertable Objects pane.

5. Click **Close**, delete the members in the report layout, and then add the same members back into the report layout from the **Insertable objects** pane.

6. Run the report.

The results appear as follows:

Sales target	2004	2005	2006	2007
Aiko Watanabe	12,260,000	9,870,300	12,648,200	4,575,700
Akemi Yamada	5,309,700	8,681,600	12,219,000	8,031,660
Alessandra Torta	7,408,000	7,996,500	8,136,100	7,529,800
Employee (by position-department)(All)	812,885,300	1,036,923,300	1,332,553,100	1,023,006,840

The rows are now returned appropriately.

Note: If you have calculations based on members in the report and their MUNs change, you will need to go into the query and manually delete the members referenced in the calculation before adding the new members to the report. This way your calculations will continue to work by referencing the new members.

7. Close all browser windows, close and save the project in Framework Manager.

Results:

You have identified how changes you make to your model can impact reports that use members. Specifically, you identified how a report can be impacted when the change you make impacts the MUN for a member in a report. This stresses the importance of using the appropriate business key right from the beginning so that reports are not broken and will help you model properly before you move to a production environment.

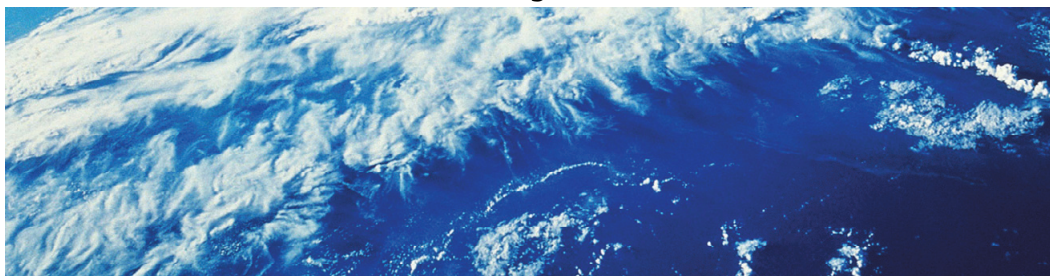
Summary

- You should now be able to:
 - describe model types and data entities
 - define members and member unique names
 - identify changes that impact a MUN



Model for Drill Through in Framework Manager

IBM Cognos BI



Business Analytics

© 2010 IBM Corporation

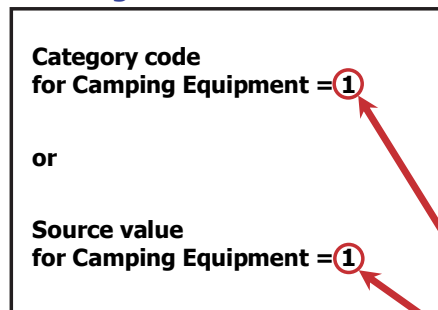
Objectives

- At the end of this module, you should be able to:
 - identify conformed values between data sources
 - define a report drill through
 - define a package-based drill through
 - identify drill-through values

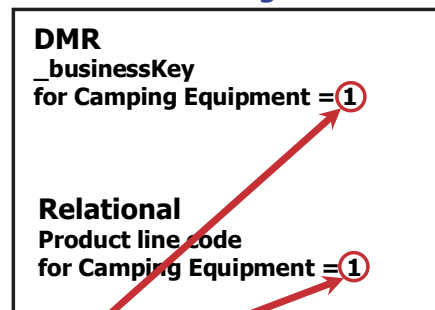
Conform Values

- To relate information between data sources, values and types must match.

IBM Cognos Transformer



Framework Manager



Conformed values for like dimensions


Cognos.
software

© 2010 IBM Corporation

If the values are not conformed for like dimensions (dimensions that represent the same data), then tasks such as creating master-detail reports and drill-through definitions will be restricted since data cannot be related on common values.

You should ensure that values are conformed and the data types match early in the modeling process to avoid future changes that impact reports and drill-through definitions.

It is ideal to have these conformed values in the data source and optimized for reporting so they are available to all modeling tools and reporting environments.

Business Analytics 


What is Drill-Through?

Source Report

Revenue	2004	2005	2006
Camping Equipment	\$20,471,328.88	\$31,373,606.46	\$37,869,055.58
Mountaineering Equipment		\$9,642,674.54	\$11,248,676.06
Personal Accessories	\$7,144,797.52	\$10,955,708.04	\$13,793,960.30
Outdoor Protection	\$1,538,456.24	\$988,230.64	\$646,428.04
Golf Equipment	\$5,597,980.86	\$9,598,268.88	\$10,709,215.84

Target Report

Year	Month	Product line	Product type	Product name	Revenue
2005	1	Mountaineering Equipment	Climbing Accessories	Firefly Charger	\$15,367.16
				Firefly Climbing Lamp	\$6,064.14
				Granite Belay	\$29,334.24
				Granite Carabiner	\$4,024.90
				Granite Chalk Bag	\$2,306.82
				Granite Pulley	\$12,538.96
				Climbing Accessories	\$69,636.22


© 2010 IBM Corporation

Drill through lets a report consumer move from one report to another to focus on a specific area of interest.

Drill through is accomplished by passing parameters from one report to another and using those parameters to filter the results. These values must be conformed to get expected results.

Business Analytics



IBM Cognos Drill-Through Definition Types

- Two commonly used types of drill-through definitions:
 - report-based
 - package-based (parameter-driven, and dynamic)
- Add parameterized filters in target reports to focus results

Cognos.
software

© 2010 IBM Corporation

There are two distinct types of drill through available within IBM Cognos. These are not the only ways of performing a drill through; however, they are the most common ways.

- A report-based drill-through definition is specific to a source report in which one or more drill paths are defined to one or more target reports. This type of definition is created in Report Studio.
- A package-based drill-through definition lets a consumer perform a drill through from any report based on a package to any other report in IBM Cognos. This type of definition is created in IBM Cognos Connection and can be either parameter-driven or dynamic in nature. When dynamic, the target report does not need filters since they will be generated dynamically at run time.

Parameterized filters ("Go To" parameter in Analysis Studio) are not required, but they are useful in returning a customized and focused view of the data based on your source drill-through value(s). Otherwise, the report is run without context from the source report.

Demo 1: Create Drill-Through Definitions

Purpose:

As a modeler, you need to understand basic drill-through functionality, so, you will create a simple report-based drill-through definition to drill from one report to another.

You will create a target report with a prompt value filter, and then a source report with a report-based drill-through definition. You will also use the target report in a package-based drill-through definition.

Components: Framework Manager, Report Studio, IBM Cognos Connection

Project: great_outdoors_warehouse

Packages: GO Data Warehouse (query), GO Data Warehouse (analysis)

Task 1. Publish package.

1. In **Framework Manager**, close any projects that may be open, and then open the **great_outdoors_warehouse** project located at **C:\Edcognos\B5152\CBI-FM-Start Files\Module 17\great_outdoors_warehouse**.

This is an untouched version of the project.

2. Publish the **GO Data Warehouse (analysis)** and **GO Data Warehouse (query)** packages.


Click **Yes**, if a warning message displays.


3. Save and close the project.

Task 2. Create the target report.

1. Launch **IBM Cognos Connection**, log on, and launch **Report Studio** selecting the **GO Data Warehouse (query)** package.
2. Create a new **List** report.
3. Expand **Sales and Marketing (query)>Sales (query)**, and then add the following items to the report:

Query Subject	Query Item
Employee by region	Region
Product	Product line Product type Product name
Sales fact	Revenue




4. In the work area, click the **Region** column header, and then Shift+click **Product type**.
5. On the toolbar, click **Group** .

- Click the **Revenue** column header, on the toolbar click **Summarize** , and then click **Total**.

The results appear as follows:

Region	Product line	Product type	Product name	Revenue
<Region>	<Product line>	<Product type>	<Product name>	<Revenue>
		<Product type> - Total		<Total(Revenue)>
		<Product type>	<Product name>	<Revenue>
		<Product type> - Total		<Total(Revenue)>
	<Product line> - Total			<Total(Revenue)>
	<Product line>	<Product type>	<Product name>	<Revenue>
		<Product type> - Total		<Total(Revenue)>
		<Product type>	<Product name>	<Revenue>
		<Product type> - Total		<Total(Revenue)>
	<Product line> - Total			<Total(Revenue)>
<Region> - Total				<Total(Revenue)>

Task 3. Create filters to be used as drill-through parameters.

- With the **Revenue** column header selected, on the toolbar, click **Filters** , then select **Edit Filters**.
- Click **Add** , click **Advanced**, and then click **OK**.
- Under the **Available Components** pane, click the **Data Items** tab .

4. In the **Available Components** pane, double-click **Region** to add it to the **Expression Definition** pane.
5. In the **Expression Definition** pane, at the end of the expression, type = **?Region?**.

The results appear as follows:

[Region] = ?Region?

You will create another filter.

6. Click **OK**, and then click **Add**.
7. Click **Advanced**, and then click **OK**.
8. Expand **Sales and Marketing (query)>Sales (query)>Product>Codes**, and then double-click **Product line code** to add it to the **Expression Definition** pane.
9. At the end of the expression, type = **?Product line code?**.

The results appear as follows:

[Sales (query)].[Product].[Product line code] = ?Product line code?

You will now make the filters optional. By making them optional, users are not forced to provide a value when running the report. This makes the report more flexible by giving consumers the choice to see all records or focus their results.

10. Click **OK**, and then under **Usage**, click **Optional**.
11. Click the **[Region] = ?Region?** filter, and then under **Usage**, click **Optional**.
12. Click **OK**.
13. From the **File** menu, click **Save**.
14. Save the report as **Revenue by Year and Product (Detail)**.

Task 4. Create the source report.



You will now create a source report based on the GO Data Warehouse (analysis) package which is modeled dimensionally (regular and measure dimensions).


1. From the **File** menu, click **New**, and then double-click **Crosstab**.
2. From the **File** menu, click **Report Package**, under **Public Folders**, click **GO Data Warehouse (analysis)**, and then click **OK**.
3. Click **OK** again.
4. Expand **Sales and Marketing (analysis)>Sales>Product>Product**, and then drag the **Product line** level to the **Rows** drop zone.
5. Expand **Employee (by region)>Employee (by region)**, and then drag the **Region** level to the **Columns** drop zone on the report.
6. Expand **Sales fact**, and then drag **Revenue** to **Measures** drop zone on the report.





The results appear as follows:

Revenue	<#Region#>	<#Region#>
<#Product line#>	<#1234#>	<#1234#>
<#Product line#>	<#1234#>	<#1234#>

Task 5. Create and test a report-based drill through.

1. Right-click one of the **Revenue** cells in the crosstab report, click **Select Fact Cells**, and then on the toolbar click **Drill-Through Definitions** .
2. Click **New Drill-Through Definition** , beside the **Report** box, click the **ellipsis**, and then double-click **Revenue By Year and Product (Detail)**.

3. Under **Parameters**, click **Edit** , and then select the **Method**, **Value**, and **Property to Pass** items as shown below:

Parameters						
Name	Type	Required	Multi-select	Method	Value	Property to Pass
Product line code	Number			Pass data item value	 Product line	 Business Key
Region	String			Pass data item value	 Region	 Member Caption

The Property to Pass item allows you to select which portion of the member (when dealing with a dimensional source) you would like to pass to the target report such as the Member Unique Name (MUN), Member Caption or Business Key. In this case you are passing the Business Key for Product line and Member Caption for Region to match the values and data types in the target relational report. These values are conformed between the two sources.


Other options include passing the Parent Unique Name (the parent MUN), Dimension Unique Name, and so on. Some of these items can be used for advanced reporting techniques. One example is to provide contextual information in the target report. You can display which dimension, hierarchy, and level a member came from in the source report. Another example is a target report with the following calculation:

```
bottomCount (#prompt('Level Name','token')#,2,[Revenue])
```

This expression will return the bottom two performing members of a level based on revenue. The level name in this case is provided through a prompt macro. You can pass the target report the Level Unique Name from the source report to determine which level will be evaluated for the bottom two members.

Please see the documentation for more details.

4. Click **OK**, and then click **OK** again.

5. On the toolbar, click **Run Report** .

The results appear as follows:

Revenue	Americas	Asia Pacific	Northern Europe	Central Europe	Southern Europe
Camping Equipment	481,445,781.04	421,639,391.62	180,851,396.88	343,645,848.36	161,454,246.13
Mountaineering Equipment	123,127,397.88	107,505,775.01	46,091,108.04	88,051,532.89	44,884,319.08
Personal Accessories	132,249,058.98	116,715,219.51	49,825,913.97	1,540,675,699.15	46,207,416.17
Outdoor Protection	23,002,647.68	19,716,018.32	8,346,431.17	17,488,870.77	7,440,328.31
Golf Equipment	217,262,995.22	193,677,873.68	84,424,300.9	153,632,833.39	77,413,364.7

6. Click **123,127,397.88** at the intersection of **Mountaineering Equipment** and **Americas** to drill through to the detail report.

The report appears as shown below:


Region	Product line	Product type	Product name	Revenue
Americas	Mountaineering Equipment	Climbing Accessories	Firefly Charger	4,578,203.51
			Firefly Climbing Lamp	2,303,453.54
			Firefly Rechargeable Battery	2,983,742.48
			Granite Belay	5,039,403.6
			Granite Carabiner	3,526,198.24
			Granite Chalk Bag	1,056,283.56
			Granite Pulley	4,256,544.54
		Climbing Accessories - Total		23,743,829.47
		Rope	Husky Rope 100	12,638,150.62
			Husky Rope 200	10,389,661.66
			Husky Rope 50	6,370,616
			Husky Rope 60	4,187,714
		Rope - Total		33,586,142.28

The report is filtered on the Mountaineering Equipment and Americas.

7. Close **IBM Cognos Viewer**, and then save the source crosstab report as **Revenue by Year and Product (Summary)**.
8. Close **Report Studio**.

Task 6. Create a package-based drill through.

You can also create a package-based drill-through definition to allow any report from a package to drill through to a target.

1. In **IBM Cognos Connection**, from the **Launch** menu, click **Drill-through Definitions**.
2. Click the **GO Data Warehouse (analysis)** package.
3. On the toolbar, click **New Drill-through Definition** .
4. In the **Name** box, type **Revenue by Year and Product (Detail)**, and then click **Next**.
5. Click **Set the target**, and then click **Select a Report**.
6. Click **Public Folders**, click **GO Data Warehouse (query)**, and then select **Revenue by Year and Product (Detail)**.
7. Click **OK**, click **Next**, and then under **Parameter mapping**, for the **Product line code** target parameter, click **map to metadata**.
8. Expand **Sales and Marketing (analysis)>Sales>Product**, click **Product line**, and then click **OK**.
9. For the **Region** target parameter, click **map to metadata**, expand **Sales and Marketing (analysis)>Sales>Employee (by region)**, and then click **Region**.
10. Click **OK**, and then under **Source metadata item properties**, select the items as shown below:

Target parameter	Type	Required	Multi-select	Source metadata item	Source metadata item properties
Product line code	Number			[Sales].[Product].[Product].[Product line] ✎ ✕	Business Key <input type="checkbox"/>
Region	Text			[Sales].[Employee (by region)].[Employee by region].[Region] ✎ ✕	Member Caption <input type="checkbox"/>

11. Click **Finish**.

Task 7. Test package-based drill through in Query Studio.

1. Launch **Query Studio**, and then select the **GO Data Warehouse (analysis)** package.
2. Expand **Sales and Marketing (analysis)>Sales**, and then add the following items to the report:

Query Subject	Query Item
Employee (by region)	Region
Product	Product line
Sales fact	Revenue

The results appear as follows:

Region	Product line	Revenue
Americas	Camping Equipment	481,445,781.04
Americas	Mountaineering Equipment	123,127,397.88
Americas	Personal Accessories	132,249,058.98
Americas	Outdoor Protection	23,002,647.68
Americas	Golf Equipment	217,262,995.22
Asia Pacific	Camping Equipment	421,639,391.62
Asia Pacific	Mountaineering Equipment	107,505,775.01
Asia Pacific	Personal Accessories	116,715,219.51
Asia Pacific	Outdoor Protection	19,716,018.32
Asia Pacific	Golf Equipment	193,677,873.68
Northern Europe	Camping Equipment	180,851,396.88
Northern Europe	Mountaineering Equipment	46,091,108.04
Northern Europe	Personal Accessories	49,825,913.97
Northern Europe	Outdoor Protection	8,346,431.17

3. Beside **Americas**, right-click **Personal Accessories**, point to **Go To**, and then click **Related Links**.
4. Click **Revenue by Year and Product (Detail)**.

The results appear as follows:

Region	Product line	Product type	Product name	Revenue
Americas	Personal Accessories	Binoculars	Seeker 35	8,874,564
			Seeker 50	6,118,790.38
			Seeker Extreme	5,696,383.38
			Seeker Mini	3,672,292.2
		Binoculars - Total		24,362,029.96
		Eyewear	Polar Extreme	999,944.29
			Polar Ice	3,420,270.02
			Polar Sports	7,987,830.88
			Polar Sun	7,644,731.52
			Polar Wave	885,230.51
		Eyewear - Total		20,938,007.22
		Knives	Bear Edge	3,861,484.85
			Bear Survival Edge	3,091,856.36
			Double Edge	3,573,659.12
			Edge Extreme	10,640,797.71
			Single Edge	13,591,099.26
		Knives - Total		34,758,897.3
		Navigation	Glacier Basic	8,796,651.34

The report is filtered on the year Personal Accessories and Americas.

5. Close **IBM Cognos Viewer**, return to **Query Studio**, and then save the report as **Module 17 Demo 1 Drill Through**.
6. Click **Return**, close **Framework Manager**, and leave **IBM Cognos Connection** open for the next demo.

Results:

By creating and testing simple drill-through definitions, you can see how parameters are passed from one report to another to focus the results of the target report.

Considerations for Drill-Through Definitions

- Things to consider before creating drill-through definitions include:
 - conformed drill-through values
 - unique business keys across all levels for IBM Cognos PowerCubes

The value passed from the source report to the target report must match in type and content.

If the values are not conformed then the filter in your target report may return no records or records that are not expected. For example, with relational to relational drill through, the value that is passed must be the same data type (integer, string, and so on) and actually exists in the target data source. If you pass a value of "Americas" to a target report that was expecting an integer, there would be a mismatch error. Also, if you passed a code of 1234 that represents the Americas but the target data source uses a code of 1 for the Americas, you would return either no results or the wrong results.

With IBM Cognos, you have the ability to use the Category code value as the member key for a PowerCube generated by IBM Cognos Transformer. If the business keys used for the category code are not unique across all levels in the PowerCube, Transformer will generate a tilde in it, such as 1~25. In these cases the value will not match the value found in the originating relational source and potentially other data sources. It is important for the category codes to be unique throughout the dimension to avoid the tilde. Because of this, it is recommended that all business keys be unique across all levels in all your data sources if you intend to use PowerCubes in drill through and master-detail operations with other sources.

When drilling from DMR or OLAP to another source, you can choose which portion of the member will be passed to match to the value used in the target report. This provides a great deal of flexibility when working with multiple data source types.

Demo 2: Identify Drill-Through Values

Purpose:

You will create a package-based drill-through definition to test values passed from an OLAP source to a relational source. By testing these values, you can obtain MUN values and compare them to the values expected in the target report.


Components: IBM Cognos Connection, Analysis Studio, Report Studio

Packages: GO Data Warehouse (query)

Task 1. Create a package-based drill through for an OLAP package.

1. In IBM Cognos Connection, from the **Launch** menu, select **Drill-through Definitions**.
2. Click **Public Folders**, and then navigate to **Samples>Cubes>Sales and Marketing (cube)**.

This package is based on an IBM Cognos PowerCube as the data source.

3. On the toolbar, click **New Drill-through Definition** .
4. In the **Name** box, type **OLAP to Relational Test**, click **Next**, and then click **Set the scope**.

You will set the scope of this drill-through definition at the Product line level of the Products dimension so that users can only drill through to the target report when at that level.

5. Expand **Products**, click **Product line**, and then click **OK**.
6. Click **Set the target**, and then click **Select a report**.
7. Click **Public Folders**, click **GO Data Warehouse (query)**, and then select **Revenue by Year and Product (Detail)**.

8. Click **OK**, and then click **Next**.
9. Under **Parameter mapping**, for the **Product line code** parameter, click **map to metadata**.
10. Expand **Products**, click **Product line**, and then click **OK**.
11. For the **Region** parameter, click **map to metadata**, expand **Retailers**, click **Region**, and then click **OK**.
12. Click **Finish**.

Task 2. Test package-based drill through in Analysis Studio.

1. From the **Launch** menu, click **Analysis Studio**, navigate to **Sample>Cubes**, and then click **Sales and Marketing (cube)**.
2. Click **Default Analysis**, and then click **OK**.
3. From the data tree, drag **Retailers** to the rows.

The results appear as follows:

Revenue	Camping Equipment	Personal Accessories	Outdoor Protection
Americas	481,445,781.04	593,696,783.38	23,00
Asia Pacific	421,639,391.62	439,800,120.35	19,71
Northern Europe	180,851,396.88	210,608,208.82	8,34
Central Europe	343,645,848.36	437,336,485.23	17,48
Southern Europe	161,454,246.13	204,231,710.00	7,44
Retailers	1,589,036,664.03	1,885,673,307.78	75,994

4. Right-click **Personal Accessories**, point to **Go To**, and then click **Related Links**.

5. Click **OLAP to Relational Test**.

By drilling through directly on Personal Accessories, you are only passing one parameter to the target report, in this case the member key value for the Product line member Personal Accessories.

The results appear as follows:

Region	Product line	Product type	Product name	Revenue
Americas	Personal Accessories	Binoculars	Seeker 35	8,874,564
			Seeker 50	6,118,790.38
			Seeker Extreme	5,696,383.38
			Seeker Mini	3,672,292.2
		Binoculars		24,362,029.96
		Eyewear	Polar Extreme	999,944.29
			Polar Ice	3,420,270.02
			Polar Sports	7,987,830.88
			Polar Sun	7,644,731.52
			Polar Wave	885,230.51
		Eyewear		20,938,007.22
		Knives	Bear Edge	3,861,484.85
			Bear Survival Edge	3,091,856.36
			Double Edge	3,573,659.12
			Edge Extreme	10,640,797.71
			Single Edge	13,591,099.26
		Knives		34,758,897.3
		Navigation	Glacier Basic	8,796,651.34

The report is filtered on the Product line code 993 which represents Personal Accessories. However, if you page through the report, you will see values are returned for all regions.

6. Close **IBM Cognos Viewer**.
7. In **Analysis Studio**, at the intersection of **Personal Accessories** and **Asia Pacific**, right-click **439,800,120.35**, point to **Go To**, and then click **Related Links**.
8. Click **View passed source values**.

This option is only available if you have been granted access. See your administrator if you require this option.

The results appear as follows:

Passed source values:		
Custom URL:		
Package-based Drill-through access:		
Package search path:		
Selection context		
Item	Display value	Use value
Revenue	439,800,120.35<A>	439,800,120.35<A>
Region	Asia Pacific	[sales_and_marketing].[Retailers].[Retailers].[Region]->:[PC].[@MEMBER].[740]
Product line	Personal Accessories	[sales_and_marketing].[Products].[Products].[Product line]->:[PC].[@MEMBER].[993]

These are all the potential values that can be passed and evaluated by the target report. The intersection you drilled through on has three elements, Product line, Region and Revenue. The target report only has optional filters on Product line code and Region. Because you did not specify an item for Source metadata item properties, the default behavior is to pass the member key to the relational source.

Notice the member key values in the MUNs. 993 for Personal Accessories and 740 for Asia Pacific. 993 is a conformed value between the OLAP source and the relational source, but 740 is not.

9. Click the arrow  beside **OLAP to Relational Test**, and then click **View Target Mapping**.

The results appear as follows:

Parameter mapping:		
Drill through definition:		
Mapped target parameters:		
Parameter name	Display value	Use value
Product line code	Personal Accessories	993
Region	Asia Pacific	740

You can see the display value of the item being passed and the actual value that will be passed to the target report. Again, the value for Product line code will work as it is conformed, but the use value for Region will not since the target report is expecting text. The value 740 will be accepted by the target report as possible text, but it does not match any values in the target data.

10. Click **OLAP to Relational Test**.

The results appear as follows:

Region	Product line	Product type	Product name	Revenue
--------	--------------	--------------	--------------	---------

The report returns no data because there are no regions called 740.

The data types do not match. There are few solutions.

- You could change the filter in the target report to accept the business key instead.
- You could change the data item being passed to caption instead of the member key in the drill-through definition.


You implement the second option.

11. Close **IBM Cognos Viewer**.

Task 3. Change the source report parameter to pass the member caption.

1. In **IBM Cognos Connection**, click **Set Properties** for **OLAP to Relational Test**, and then click the **Target** tab.
2. Under **Parameter mapping**, in the **Region** row, change **Source metadata item properties** to **Member Caption**.
3. Click **OK**.

Task 4. Retest the drill-through definition in Analysis Studio.

1. In **Analysis Studio**, at the intersection of **Personal Accessories** and **Asia Pacific**, right-click **439,800,120.35**, point to **Go To**, and then click **Related Links**.
2. Click the arrow  beside **OLAP to Relational Test**, and then click **View Target Mapping**.

The results appear as follows:

Parameter mapping:		
Drill through definition:		
Mapped target parameters:		
Parameter name	Display value	Use value
Product line code	Personal Accessories	993
Region	Asia Pacific	Asia Pacific

The Use value for Region is now text and a compatible value with the relational target report.

3. Click **OLAP to Relational Test**.

The results appear as follows:

Region	Product line	Product type	Product name	Revenue
Asia Pacific	Personal Accessories	Binoculars	Seeker 35	8,059,758.48
			Seeker 50	5,499,440.04
			Seeker Extreme	5,111,979.2
			Seeker Mini	3,223,058.37
		Binoculars - Total		21,894,236.09
		Eyewear	Polar Extreme	874,608.04
			Polar Ice	3,082,386.12
			Polar Sports	7,280,025.69
			Polar Sun	6,886,928.26
			Polar Wave	782,349.02
		Eyewear - Total		18,906,297.13
		Knives	Bear Edge	3,330,974.78
			Bear Survival Edge	2,671,570.34
			Double Edge	3,095,834.16
			Edge Extreme	9,232,660.21
			Single Edge	11,764,048.81
		Knives - Total		30,095,088.3
		Navigation	Glacier Basic	7,577,654.38

The report is now filtered on both Personal Accessories and Asia Pacific because all drill-through values are conformed.

4. Close all **browser** windows.

Results:

By testing values passed from an OLAP source to a relational source, you observed techniques to obtain MUN values and compared them to the values expected in the target report.

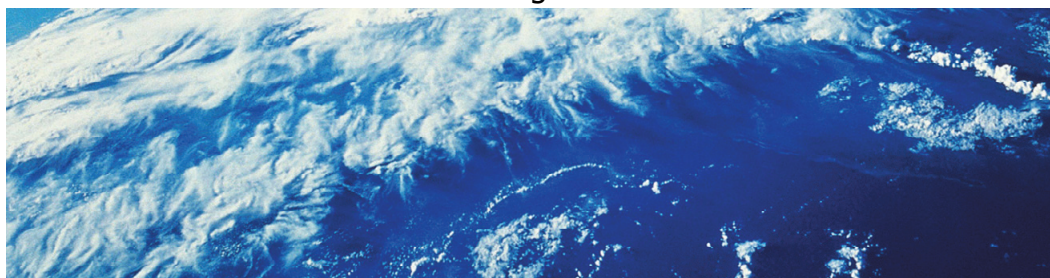
Summary

- You should be able to:
 - identify conformed values between data sources
 - define a report drill through
 - define a package-based drill through
 - identify drill-through values



Advanced Generated SQL Concepts and Complex Queries

IBM Cognos BI




Business Analytics

© 2010 IBM Corporation

Objectives

- At the end of this module, you should be able to identify:
 - governors that affect SQL generation
 - stitch query SQL
 - conformed and non-conformed dimensions in generated SQL
 - multi-fact/multi-grain stitch query SQL
 - variances in Report Studio generated SQL
 - dimensionally modeled relational SQL generation
 - cross join SQL
 - various results sets for multi-fact queries

Business Analytics


Explore SQL Generation

```


select
  coalesce(D2.Year1,D3.Year1) as Year1,
  D2.Revenue as Revenue,
  D3.Sales_Target as Sales_Target
from
  (select
    Time_Dimension.Year1 as Year1,
    XSUM(Sales_Fact.Revenue for
      Time_Dimension.Year1) as Revenue
    from
      .....
    where
      (Time_Dimension.Day_Date =
        Sales_Fact.Order_Date)
    group by
      Time_Dimension.Year1
  ) D2

```

```

full outer join
  (select
    Time_Dimension.Year1 as Year1,
    XSUM(Sales_Target.Sales_Target
      for Time_Dimension.Year1) as
    Sales_Target
  from
    .....
  where
    ((Time_Dimension.Year1 =
      Sales_Target.Sales_Year) and
    (Time_Dimension.Month1 =
      Sales_Target.Sales_Period))
  group by
    Time_Dimension.Year1
  ) D3
on (D2.Year1 = D3.Year1)

```


© 2010 IBM Corporation

Confusion is a common reaction to the SQL illustrated in the slide example.

Frequently asked questions are:

- What is the coalesce function?
- Why do I see the same columns being selected in two different derived tables?
- Why do I see a full outer join?
- What does the XSUM function do?

These questions and others will be answered throughout this module as you explore complex Cognos generated SQL.

Governors that affect SQL Generation

- Outer Joins (Allow, Deny)
- Cross-Product Joins (Allow, Deny)
- Shortcut Processing (Automatic, Explicit)
- SQL Join Syntax (Explicit, Implicit)
- Grouping of Measure Attributes (Enable, Disable)
- SQL Generation for Level Attributes (Group by, Minimum)
- SQL Generation for Determinant Attributes (Group by, Minimum)
- SQL Parameter Syntax (Marker, Literal)
- Use WITH clause when generating SQL (Yes, No)

There are several project governors that can limit queries and affect the SQL generated at run time. Some governors are discussed in more detail in another module. We will not go into detail for each governor setting listed above since some are used in rare cases. Please refer to the documentation for details on each governor setting. We will look at a couple in detail.

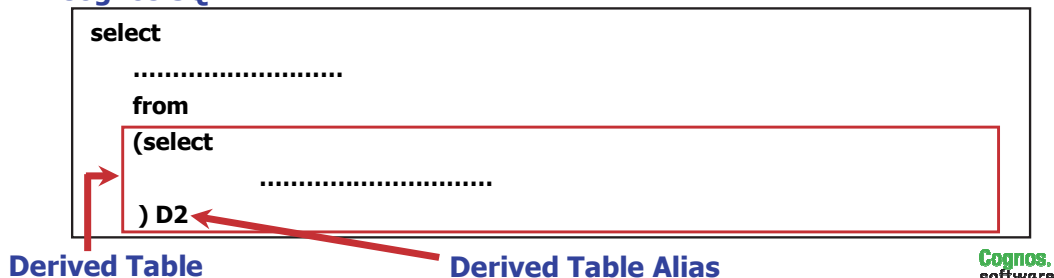
The SQL Join Syntax governor controls how SQL is generated for inner joins. Selecting Explicit will generate INNER JOIN syntax, and selecting Implicit will use WHERE syntax.

The Use WITH clause when generating SQL governor lets you choose to use the WITH clause with Cognos SQL if your data source supports it.

Derived Tables

- Derived tables:
 - are aliased sub-selects
 - enable developers to see values being returned without viewing complex SQL
 - are generated in both Cognos SQL and Native SQL

Cognos SQL



Before examining more complex stitch query SQL, let's take a moment to review derived tables.

A derived table retrieves a record set that fulfills the requirements of the parent query.


Derived Tables (cont'd)

<pre> select coalesce(D2.Year1,D3.Year1) as Year1, D2.Revenue as Revenue, D3.Sales_Target as Sales_Target from (select Time_Dimension.Year1 as Year1, XSUM(Sales_Fact.Revenue for Time_Dimension.Year1) as Revenue from where (Time_Dimension.Day_Date = Sales_Fact.Order_Date) group by Time_Dimension.Year1) D2 </pre>	<pre> full outer join (select Time_Dimension. Year1 as Year1, XSUM(Sales_Target.Sales_Target for Time_Dimension.Year1) as Sales_Target from where ((Time_Dimension.Year1 = Sales_Target.Sales_Year) and (Time_Dimension.Month1 = Sales_Target.Sales_Period)) group by Time_Dimension.Year1) D3 on (D2.Year1 = D3.Year1) </pre>
---	---

Derived tables use alias names that make it easy to see from which query the projected items come from.

In the slide example, we have two derived tables, D2 and D3, which achieve the final projections list.

The derived table alias names are also used in the join statement.

Business Analytics


Identify Stitch Query SQL

```

select
  coalesce(D2.Year1,D3.Year1) as Year1,
  D2.Revenue as Revenue,
  D3.Sales_Target as Sales_Target
from
  (select
    Time_Dimension.Year1 as Year1,
    XSUM(Sales_Fact.Revenue for
      Time_Dimension.Year1) as Revenue
    from
      .....
    where
      (Time_Dimension.Day_Date =
        Sales_Fact.Order_Date)
    group by
      Time_Dimension.Year1
  ) D2


```

full outer join

```

(select
  Time_Dimension. Year1 as Year1,
  XSUM(Sales_Target.Sales_Target
    for Time_Dimension.Year1) as
    Sales_Target
  from
    .....
  where
    ((Time_Dimension.Year1 =
      Sales_Target.Sales_Year) and
    (Time_Dimension.Month1 =
      Sales_Target.Sales_Period))
  group by
    Time_Dimension.Year1
) D3
on (D2.Year1 = D3.Year1)

```



© 2010 IBM Corporation

Stitch queries are used to achieve predictable results for multi-fact queries.

There are three essential components of a stitch query:

- coalesce function
- full outer join
- multiple queries that query some of the same information

What is a Coalesce Function?

```
select
  coalesce(D2.DAY_DATE,D3.DAY_DATE) as DAY_DATE,
  coalesce(D2.ORDER_METHOD,D3.ORDER_METHOD) as ORDER_METHOD,
```

Sales Fact

DAY_DATE	ORDER_METHOD	SALE_TOTAL
01/01/2005	E-mail	\$10
01/02/2005	Telephone	\$25
01/03/2005	Web	\$40
01/04/2005	E-mail	\$20

Returned Items Fact

DAY_DATE	ORDER_METHOD	RETURN_QUANTITY
01/01/2005	E-mail	2
01/02/2005	Telephone	4
01/10/2005	Fax	15
01/11/2005	Sales visit	1

Report Set

DAY_DATE	ORDER_METHOD	SALE_TOTAL	RETURN_QUANTITY
01/01/2005	E-mail	\$10	2
01/02/2005	Telephone	\$25	4
01/03/2005	Web	\$40	
01/04/2005	E-mail	\$20	
01/10/2005	Fax		15
01/11/2005	Sales visit		1

A coalesce function:

- merges query items that exist on multiple sides of the query
- indicates that a query item is part of a conformed dimension

Anything included in the report as a column, filter, or prompt can be treated as a conformed dimension, if it is common to the fact items in the query, and may appear as a coalesce.

Non-Conformed Dimensions in Generated SQL

```

select
  coalesce(D2.DAY_DATE,D3.DAY_DATE) as DAY_DATE,
  coalesce(D2.ORDER_METHOD,D3.ORDER_METHOD) as ORDER_METHOD,
  D3.REASON_DESCRIPTION as REASON_DESCRIPTION,
  D2.SALE_TOTAL as SALE_TOTAL,
  D3.RETURN_QUANTITY as RETURN_QUANTITY
from
  (select
    Time_Dimension.DAY_DATE as DAY_DATE,
    Order_Method_Dimension.ORDER_METHOD_EN as ORDER_METHOD,
    Return_Reason_Dimension.REASON_DESCRIPTION REASON_DESCRIPTION,
    .....
  group by
    Time_Dimension.DAY_DATE,
    Order_Method_Dimension.ORDER_METHOD_EN,
    Return_Reason_Dimension.REASON_DESCRIPTION
  ) D3
full outer join
  (select
    .....
  ) D2
on ((D2.DAY_DATE = D3.DAY_DATE) and (D2.ORDER_METHOD = D3.ORDER_METHOD))

```

**Non-
conformed
dimension**

**Found only
in query of
related
fact**

**Cognos.
software**

© 2010 IBM Corporation

Non-conformed dimensions will not use a coalesce function and only show up in the derived table of the fact to which they are related.

If you are using what you expect to be a conformed dimension in a multi-fact query and no coalesce function is generated, you should investigate your model. Ensure no query path has been missed or that the IBM Cognos query engine is not identifying the dimension as a fact based on cardinality.

What is an RSUM(1 asc local) as sc?

- IBM Cognos is generating a stitch column that will be used to stitch queries together locally.
- found in multi-fact queries when automatic summarization is not enabled, when there is no common level of granularity, and when:
 - at least one conformed dimension is present

RSUM(1 for Order_Method_Dimension.ORDER_METHOD_KEY order by Order_Method_Dimension.ORDER_METHOD_KEY asc local) as sc1

- no conformed dimensions are present

RSUM(1 order by Sales_Target_Fact.SALES_TARGET asc local) as sc

If automatic aggregation is not enabled and at least one conformed dimension is present, you will see the same stitch column generated in both derived tables of a stitch query. This stitch column takes a common key(s) from the conformed dimension(s) between the two queries and sorts it ascending locally on the IBM Cognos server. These columns and others then merge the two result sets. The fact values in the fact columns will be related to the conformed dimension but not necessarily to each other.

If automatic aggregation is not enabled and there are no conformed dimensions present, IBM Cognos will attempt to generate a stitch column by selecting a column from each query and using it to create unique values that will merge the queries. There are no definite relationships between the facts.

In either case RSUM(1....asc local) as sc is cause for investigation to ensure the correct results are returned.

What is an RSUM(1 asc local) as sc? (cont'd)

- The following syntax indicates that a conformed dimension has not been included in a multi-fact query.

Stitch column from Query 1 of stitch query

RSUM(1 order by ORDER_DETAILS.ORDER_DETAIL_CODE asc local) as sc

.....
full outer join
.....

RSUM(1 order by PRODUCT_FORECAST.PRODUCT_NUMBER asc local) as sc

Stitch column from Query 2 of stitch query

.....
on (Query 1.sc = Query 2.sc)

Cognos.
software

© 2010 IBM Corporation

Each query uses a different column to generate the stitch column that in most cases, returns unrelated results.

Why Do I See XSUM?

- For readability, Cognos SQL uses windowed aggregates.
- Cognos SQL:
 - **XSUM**(Sales_Fact.SALE_TOTAL for Time_Dimension.MONTH_KEY) as SALE_TOTAL
- Native SQL:
 - **sum**("Sales_Fact"."SALE_TOTAL") AS "SALE_TOTAL"

XSUM in Cognos SQL indicates a windowed aggregate, in which you can see what value is being aggregated and to what level(s).

The X in XSUM stands for extended, which indicates that the overall total for each row of a particular grouping will be calculated and retrieved.

Demo 1: Identify Stitch Queries in Generated SQL

Purpose:

When running multi-fact queries, you must identify the components of the generated SQL. Understanding the patterns of correctly generated stitch queries will let you effectively troubleshoot improperly constructed queries. To that end, you will test various multi-fact query scenarios and explore the generated SQL.

Component: Framework Manager

Project: GO Operational

Task 1. Test fact queries individually.

Before testing a multi-fact query, you will test each fact query to examine the generated SQL.

1. In **Framework Manager**, close any projects that may be open, and then open the **GO Operational** project located at **C:\Edcognos\B5152\CBI-FM-Start Files\Module 18\GO Operational**.
2. If prompted, log in as User ID **admin**, and Password **Education1!**.
3. In the **Project Viewer**, expand **GO Operational Model>Consolidation View**.
4. Test the following items together:

Query Subject	Query Item
Time	Month
Sales Fact	Revenue

The Test Results dialog appears. You will view the generated SQL before applying Auto Sum.

- Click the **Query Information** tab.

The results appear as follows:

```

Cognos SQL
select
    TIME_DIMENSION.MONTH_EN as Month1,
    Sales_Fact.Revenue as Revenue
from
    GOSALES.GOSALES.gosales.TIME_DIMENSION TIME_DIMENSION,
    (select
        ORDER_HEADER.ORDER_DATE as ORDER_DATE,
        (ORDER_DETAILS.QUANTITY * ORDER_DETAILS.UNIT_SALE_PRICE) as Revenue
    from
        GOSALES.GOSALES.gosales.ORDER_HEADER ORDER_HEADER,
        GOSALES.GOSALES.gosales.ORDER_DETAILS ORDER_DETAILS
    where
        (ORDER_HEADER.ORDER_NUMBER = ORDER_DETAILS.ORDER_NUMBER)
    ) Sales_Fact
where
    (TIME_DIMENSION.DAY_DATE = Sales_Fact.ORDER_DATE)

```

You see a basic query where the TIME_DIMENSION dimension and Sales Fact query subjects are joined. A derived table is generated in the Cognos SQL for Sales Fact since the Revenue column you selected in the query is based on a calculation. Again, Cognos SQL is more verbose. The native SQL does not require a derived table for this basic calculation based on columns from the same table.

Note: You see Month1 in the generated SQL as opposed to Month because month is a reserved word. Therefore, a 1 is appended to avoid any conflicts.

- Click the **Test** tab, select the **Auto Sum** check box, and then click **Test Sample**.

7. Click the **Query Information** tab.

The results appear as follows:

Cognos SQL	
<pre> select TIME_DIMENSION.MONTH_EN as Month1, XSUM(Sales_Fact.Revenue for TIME_DIMENSION.MONTH_EN) as Revenue from GOSALES.GOSALES.gosales.TIME_DIMENSION TIME_DIMENSION, (select ORDER_HEADER.ORDER_DATE as ORDER_DATE, (ORDER_DETAILS.QUANTITY * ORDER_DETAILS.UNIT_SALE_PRICE) as Revenue from GOSALES.GOSALES.gosales.ORDER_HEADER ORDER_HEADER, GOSALES.GOSALES.gosales.ORDER_DETAILS ORDER_DETAILS where (ORDER_HEADER.ORDER_NUMBER = ORDER_DETAILS.ORDER_NUMBER)) Sales_Fact where (TIME_DIMENSION.DAY_DATE = Sales_Fact.ORDER_DATE) group by TIME_DIMENSION.MONTH_EN </pre>	
Native SQL	
<pre> select "TIME_DIMENSION"."MONTH_EN" AS "Month1", sum("coguda11"."QUANTITY" * "coguda11"."UNIT_SALE_PRICE") AS "Revenue" from "GOSALES"."gosales"."TIME_DIMENSION" "TIME_DIMENSION", "GOSALES"."gosales"."ORDER_HEADER" "coguda10", "GOSALES"."gosales"."ORDER_DETAILS" "coguda11" where "coguda10"."ORDER_NUMBER" = "coguda11"."ORDER_NUMBER" and "TIME_DIMENSION"."DAY_DATE" = "coguda10"."ORDER_DATE" group by "TIME_DIMENSION"."MONTH_EN" </pre>	

Now you see that XSUM has been applied in the Cognos SQL to aggregate Revenue to the month level. The sum function is used in the native SQL.

8. Click **Close**.
9. In the **Project Viewer**, test the following items together from the **Consolidation View**:

Query Subject	Query Item
Time	Month
Sales Target Fact	Sales Target

10. Click the **Query Information** tab.

The results appear as follows:

```

Cognos SQL
select
    TIME_DIMENSION.MONTH_EN as Month1,
    SALES_TARGET.SALES_TARGET as Sales_Target
from
    (select
        TIME_DIMENSION.CURRENT_YEAR as CURRENT_YEAR,
        TIME_DIMENSION.CURRENT_MONTH as CURRENT_MONTH,
        XMIN(TIME_DIMENSION.MONTH_EN for TIME_DIMENSION.CURRENT_YEAR,TIME_DIMENSION.CUR
    from
        GOSALES.GOSALES.gosales.TIME_DIMENSION TIME_DIMENSION
    group by
        TIME_DIMENSION.CURRENT_YEAR,
        TIME_DIMENSION.CURRENT_MONTH
    ) TIME_DIMENSION,
    GOSALES.GOSALES.gosales.SALES_TARGET SALES_TARGET
where
    ((TIME_DIMENSION.CURRENT_YEAR = SALES_TARGET.SALES_YEAR) and (TIME_DIMENSION.CURRENT_MO

```

Because Sales Target rolls up to the month level and not the day level and you have specified determinants on the TIME_DIMENSION, an XMIN function and a group by clause is generated in the Cognos SQL. The XMIN function in the Cognos SQL (min in the native SQL) ensures that only one month value is returned for each month. The determinant for the month level specifies a multi-part key, which is why the derived table for TIME_DIMENSION uses a group by clause on CURRENT_YEAR and CURRENT_MONTH. This prevents double-counting for Sales Target because the values are not aggregated for every day in the month, but rather at the grouped month level.

Without the determinants, the generated SQL would not include the XMIN function or a grouping on the keys. You will test this in the next few steps.

11. Click **Close**, expand **Foundation Objects View>gosales**, and then double-click **TIME_DIMENSION**.
12. Click the **Determinants** tab, delete all the determinants except for the **Day** determinant, and then click **OK**.

13. Test the following query items in the **Consolidation View**:

Query Subject	Query Item
Time	Month
Sales Target Fact	Sales Target

Remember, the items in the Consolidation View are based on items in the Foundation Objects View. You are testing the objects that authors will use.

14. Click the **Query Information** tab.

The results appear as follows:

Cognos SQL
<pre>select TIME_DIMENSION.MONTH_EN as Month1, SALES_TARGET.SALES_TARGET as Sales_Target from GOSALES.GOSALES.gosales.TIME_DIMENSION TIME_DIMENSION, GOSALES.GOSALES.gosales.SALES_TARGET SALES_TARGET where ((TIME_DIMENSION.CURRENT_YEAR = SALES_TARGET.SALES_YEAR) and (TIME</pre>
Native SQL
<pre>select "TIME_DIMENSION"."MONTH_EN" AS "Month1", "SALES_TARGET"."SALES_TAR "GOSALES"."gosales"."TIME_DIMENSION" "TIME_DIMENSION", "GOSALES"."gosales "TIME_DIMENSION"."CURRENT_YEAR" = "SALES_TARGET"."SALES_YEAR" and "TIME_D "SALES_TARGET"."SALES_PERIOD"</pre>

The XMIN function and group by clause is no longer present. Now, if you aggregated Sales Target, each value would be double-counted, once for every day in the month for which it is associated.

You can verify the effect of double-counting by retesting with Auto Sum enabled. Take note of the totals and then compare them to the values seen in the following steps when determinants are returned.

15. Click **Close**, and then from the **Edit** menu, click **Undo Edit Definition**.

16. Retest the same query items with **Auto Sum** enabled, and then click the **Query Information** tab.

The results appear as shown below:

Cognos SQL
<pre> select TIME_DIMENSION.MONTH_EN as Month1, XSUM(SALES_TARGET.SALES_TARGET for TIME_DIMENSION.MONTH_EN) as Sales_Targ from (select TIME_DIMENSION.CURRENT_YEAR as CURRENT_YEAR, TIME_DIMENSION.CURRENT_MONTH as CURRENT_MONTH, XMIN(TIME_DIMENSION.MONTH_EN for TIME_DIMENSION.CURRENT_YEAR,TIME_DIM from GOSALES.GOSALES.gosales.TIME_DIMENSION TIME_DIMENSION group by TIME_DIMENSION.CURRENT_YEAR, TIME_DIMENSION.CURRENT_MONTH) TIME_DIMENSION, GOSALES.GOSALES.gosales.SALES_TARGET SALES_TARGET where ((TIME_DIMENSION.CURRENT_YEAR = SALES_TARGET.SALES_YEAR) and (TIME_DIMENSION. group by TIME_DIMENSION.MONTH_EN </pre>
Native SQL
<pre> select "TIME_DIMENSION"."MONTH_EN" AS "Month1", sum("SALES_TARGET"."SALES_TARGET") A "TIME_DIMENSION"."CURRENT_YEAR" AS "CURRENT_YEAR", "TIME_DIMENSION"."CURRENT_MONTH" ("TIME_DIMENSION"."MONTH_EN") AS "MONTH_EN" from "GOSALES"."gosales"."TIME_DIMENSION </pre>

Now you see that XSUM has been applied in the Cognos SQL to aggregate Sales Target to the month level. The sum function is used in the native SQL.

Again, if you examine the sales target values, you will see they appear correctly now that determinants have been re-applied.

17. Click **Close**.

Task 2. Test multi-fact/multi-grain query.

Now that you have seen how the generated SQL appears for each of the fact queries individually, you will test them together. Not only is this test for multiple facts, it also illustrates multiple levels of granularity. Revenue rolls up to the day level and Sales Target rolls up to the month level.

1. In the **Project Viewer**, in the **Consolidation View**, test the following items together:

Query Subject	Query Item
Time	Month
Sales Fact	Revenue
Sales Target Fact	Sales Target

Due to the large data set, this query may take some time.

2. Click the **Query Information** tab.

The results appear as follows:

```

Cognos SQL
select
    coalesce(D2.Month2,D3.Month2) as Month1,
    D2.Revenue as Revenue,
    D3.Sales_Target as Sales_Target
from
    (select
        TIME_DIMENSION.DAY_KEY as sc,
        TIME_DIMENSION.MONTH_EN as Month2,
        Sales_Fact.Revenue as Revenue,
        RSUM(1 for TIME_DIMENSION.DAY_KEY order by TIME_DIMENSION.DAY_KEY asc local)
    from
        GOSALES.GOSALES.gosales.TIME_DIMENSION TIME_DIMENSION,
        (select
            ORDER_HEADER.ORDER_DATE as ORDER_DATE,
            (ORDER_DETAILS.QUANTITY * ORDER_DETAILS.UNIT_SALE_PRICE) as Revenue
        from
            GOSALES.GOSALES.gosales.ORDER_HEADER ORDER_HEADER,
            GOSALES.GOSALES.gosales.ORDER_DETAILS ORDER_DETAILS
        where
            (ORDER_HEADER.ORDER_NUMBER = ORDER_DETAILS.ORDER_NUMBER)
        ) Sales_Fact
    where
        (TIME_DIMENSION.DAY_DATE = Sales_Fact.ORDER_DATE)
    order by
        sc asc
    ) D2
full outer join
(select
    TIME_DIMENSION.DAY_KEY as sc,
    TIME_DIMENSION.MONTH_EN as Month2,
    SALES_TARGET.SALES_TARGET as Sales_Target,
    RSUM(1 for TIME_DIMENSION.DAY_KEY order by TIME_DIMENSION.DAY_KEY asc local)
from
    GOSALES.GOSALES.gosales.TIME_DIMENSION TIME_DIMENSION,
    GOSALES.GOSALES.gosales.SALES_TARGET SALES_TARGET
where
    ((TIME_DIMENSION.CURRENT_YEAR = SALES_TARGET.SALES_YEAR) and (TIME_DIMENSION.CU
order by
    sc asc
) D3
on ((D2.sc = D3.sc) and (D2.sc4 = D3.sc4))
    
```

The generated Cognos SQL presents a coalesce function, as expected, because Time is a conformed dimension between Sales Fact and Sales Target Fact. Auto Sum is not enabled, so there are two facts without a common level of granularity between them. Revenue rolls up to the day level and Sales Target rolls up to the month level. This is why we see the RSUM(1....asc local) as sc syntax in both derived tables, D2 and D3, which are joined together by a full outer join. The full outer join uses two stitch columns, sc, and sc4, to merge the two record sets together.

The native SQL does not present a full outer join. It is actually two separate queries sent to the database, since we are asking for local processing in our RSUM functions. Therefore, the merging is done locally. The merged results for Revenue and Sales Target will be related to Month, but not necessarily to each other.

3. Click the **Test** tab, select **Auto Sum**, and then click **Test Sample**.

4. Click the **Query Information** tab.

The results appear as shown below:

```

Cognos SQL
select
    coalesce(D2.Month1,D3.Month1) as Month1,
    D2.Revenue as Revenue,
    D3.Sales_Target as Sales_Target
from
    (select
        TIME_DIMENSION.MONTH_EN as Month1,
        XSUM(Sales_Fact.Revenue for TIME_DIMENSION.MONTH_EN ) as Revenue
    from
        GOSALES.GOSALES.gosales.TIME_DIMENSION TIME_DIMENSION,
        (select
            ORDER_HEADER.ORDER_DATE as ORDER_DATE,
            (ORDER_DETAILS.QUANTITY * ORDER_DETAILS.UNIT_SALE_PRICE) as Revenue
        from
            GOSALES.GOSALES.gosales.ORDER_HEADER ORDER_HEADER,
            GOSALES.GOSALES.gosales.ORDER_DETAILS ORDER_DETAILS
        where
            (ORDER_HEADER.ORDER_NUMBER = ORDER_DETAILS.ORDER_NUMBER)
        ) Sales_Fact
    where
        (TIME_DIMENSION.DAY_DATE = Sales_Fact.ORDER_DATE)
    group by
        TIME_DIMENSION.MONTH_EN
    ) D2
full outer join
(select
    TIME_DIMENSION.MONTH_EN as Month1,
    XSUM(SALES_TARGET.SALES_TARGET for TIME_DIMENSION.MONTH_EN ) as Sales_Target
from
    (select
        TIME_DIMENSION.CURRENT_YEAR as CURRENT_YEAR,
        TIME_DIMENSION.CURRENT_MONTH as CURRENT_MONTH,
        XMIN(TIME_DIMENSION.MONTH_EN for TIME_DIMENSION.CURRENT_YEAR,TIME_DIMEN
    from
        GOSALES.GOSALES.gosales.TIME_DIMENSION TIME_DIMENSION
    group by
        TIME_DIMENSION.CURRENT_YEAR,
        TIME_DIMENSION.CURRENT_MONTH
    ) TIME_DIMENSION,
    GOSALES.GOSALES.gosales.SALES_TARGET SALES_TARGET
    where
        ((TIME_DIMENSION.CURRENT_YEAR = SALES_TARGET.SALES_YEAR) and (TIME_DIMENSION.CU
    group by
        TIME_DIMENSION.MONTH_EN
    ) D3
    on (D2.Month1 = D3.Month1)

```

If you examine the D2 and D3 derived tables you will see that they represent the same generated SQL you saw when you tested the fact queries individually. However, now they are a part of a larger query that will merge them. There is also a common level of granularity between the queries since both are now aggregated to and grouped by the Month level. If you examine the native SQL we will see that rather than two separate queries, IBM Cognos is now sending a single query and requesting that the database process the full outer join.

5. Click **Close**.

Task 3. Identify improperly formed multi-fact queries.

1. In the **Project Viewer**, in the **Consolidation View**, test the following items together:

Query Subject	Query Item
Order Method	Order Method
Sales Fact	Revenue
Sales Target Fact	Sales Target

2. Click the **Query Information** tab.

The results appear as follows:

```

Cognos SQL
select
    D2.Order_Method as Order_Method,
    D2.Revenue as Revenue,
    D3.Sales_Target as Sales_Target
from
    (select
        ORDER_METHOD.ORDER_METHOD_EN as Order_Method,
        Sales_Fact.Revenue as Revenue,
        RSUM(1 order by ORDER_METHOD.ORDER_METHOD_EN asc local) as sc
    from
        GOSALES.GOSALES.gosales.ORDER_METHOD ORDER_METHOD,
        (select
            ORDER_HEADER.ORDER_METHOD_CODE as ORDER_METHOD_CODE,
            (ORDER_DETAILS.QUANTITY * ORDER_DETAILS.UNIT_SALE_PRICE) as
        from
            GOSALES.GOSALES.gosales.ORDER_HEADER ORDER_HEADER,
            GOSALES.GOSALES.gosales.ORDER_DETAILS ORDER_DETAILS
        where
            (ORDER_HEADER.ORDER_NUMBER = ORDER_DETAILS.ORDER_NUMBER)
        ) Sales_Fact
    where
        (ORDER_METHOD.ORDER_METHOD_CODE = Sales_Fact.ORDER_METHOD_CODE)
    order by
        Order_Method asc
    ) D2
    full outer join
    (select
        SALES_TARGET.SALES_TARGET as Sales_Target,
        RSUM(1 order by SALES_TARGET.SALES_TARGET asc local) as sc
    from
        GOSALES.GOSALES.gosales.SALES_TARGET SALES_TARGET
    order by
        Sales_Target asc
    ) D3
    on (D2.sc = D3.sc)

```

You can immediately tell that this query is suspect by the absence of the coalesce function. When you author multi-fact queries, you must include at least one conformed dimension. Other indicators are the generated stitch columns highlighted above. Each one uses a different column to generate a key that will merge the queries.

3. Click the **Test** tab, select **Auto Sum**, and then click **Test Sample**.

The results appear as follows:

	Order Method	Revenue	Sales Target
	Sales visit	310194834	4205368540
	E-mail	179843044.16	4205368540
	Mail	46091338.97	4205368540
	Special	27351320.25	4205368540
	Telephone	340985781.06	4205368540
	Fax	70073542.01	4205368540
	Web	3712235908.4	4205368540

Sales Target repeats the overall total for each row.

- Click the **Query Information** tab.

The results appear as follows:

```

Cognos SQL
select
    D2.Order_Method as Order_Method,
    D2.Revenue as Revenue,
    D3.Sales_Target as Sales_Target
from
    (select
        ORDER_METHOD.ORDER_METHOD_EN as Order_Method,
        XSUM(Sales_Fact.Revenue for ORDER_METHOD.ORDER_METHOD_EN ) as Revenue
    from
        GOSALES.GOSALES.gosales.ORDER_METHOD ORDER_METHOD,
        (select
            ORDER_HEADER.ORDER_METHOD_CODE as ORDER_METHOD_CODE,
            (ORDER_DETAILS.QUANTITY * ORDER_DETAILS.UNIT_SALE_PRICE) as Revenue
        from
            GOSALES.GOSALES.gosales.ORDER_HEADER ORDER_HEADER,
            GOSALES.GOSALES.gosales.ORDER_DETAILS ORDER_DETAILS
        where
            (ORDER_HEADER.ORDER_NUMBER = ORDER_DETAILS.ORDER_NUMBER)
        ) Sales_Fact
    where
        (ORDER_METHOD.ORDER_METHOD_CODE = Sales_Fact.ORDER_METHOD_CODE)
    group by
        ORDER_METHOD.ORDER_METHOD_EN
    ) D2,
    (select distinct
        XSUM(SALES_TARGET.SALES_TARGET ) as Sales_Target
    from
        GOSALES.GOSALES.gosales.SALES_TARGET SALES_TARGET
    ) D3
  
```

Again, the query can still be identified as an improper attempt to query multiple facts. There is no coalesce function, and you are sending two separate queries, one for each fact. The first derived table, which queries the Revenue fact, appears to be correct. But the second derived table, which queries the Sales Target fact, is simply requesting a distinct overall total. This value will be repeated for each row returned by the first derived table.

- Click **Close**, and leave Framework Manager open.

Results:

By testing various fact and multi-fact queries, you identified traits and patterns in the generated SQL, which indicate correctly modeled query subjects. You also examined and identified patterns for improperly formed multi-fact queries.

Workshop 1: Reverse Engineer a Framework Manager Model from Generated Cognos SQL

You have just been handed some generated SQL from a report author who is expressing concern regarding the SQL. This SQL uses explicit join syntax (INNER JOIN) rather than implicit join syntax (WHERE clauses). Review the SQL and try to reverse engineer the model based on the query subjects used to author the report. During this process, take note of some of the unexpected SQL and explain it.

Write notes in the margin explaining the various portions of the SQL. Use the blank space provided to draw the report that this SQL created, and a free-hand diagram of the portion of the model representing the queries used in the report. Be sure to include the cardinalities in the diagram.

```
select
  coalesce(D2.MONTH1,D3.MONTH1) as MONTH1,
  coalesce(D2.ORDER_METHOD,D3.ORDER_METHOD) as
    ORDER_METHOD,
  D3.REASON_DESCRIPTION as REASON_DESCRIPTION,
  D2.QUANTITY as QUANTITY,
  D3.RETURN_QUANTITY as RETURN_QUANTITY

from

(select

  Time_Dimension.MONTH1 as MONTH1,
  Order_Method_Dimension.ORDER_METHOD as
    ORDER_METHOD,
  Return_Reason_Dimension.REASON_DESCRIPTION as
    REASON_DESCRIPTION,
  XSUM(Returned_Items_Fact.RETURN_QUANTITY for
  Time_Dimension.MONTH1,Order_Method_Dimension.ORDER_M
  ETHOD,Return_Reason_Dimension.REASON_DESCRIPTION )
  as RETURN_QUANTITY

from
```

```

(select
    Time_Dimension.DAY_KEY as DAY_KEY,
    Time_Dimension.MONTH_EN as MONTH1
from
    go_data_warehouse.GOSLDW.dbo.TIME_DIMENSION
    Time_Dimension) Time_Dimension

join

go_data_warehouse.GOSLDW.dbo.RETURNED_ITEMS_FACT
Returned_Items_Fact

on (Time_Dimension.DAY_KEY =
    Returned_Items_Fact.DAY_KEY)

join

(select
    Order_Method_Dimension.ORDER_METHOD_KEY
    as ORDER_METHOD_KEY,
    Order_Method_Dimension.ORDER_METHOD_EN
    as ORDER_METHOD

from
    go_data_warehouse.GOSLDW.dbo.ORDER_METHOD_DIMENSION
    Order_Method_Dimension) Order_Method_Dimension

on (Order_Method_Dimension.ORDER_METHOD_KEY =
    Returned_Items_Fact.ORDER_METHOD_KEY)

join

(select
    Return_Reason_Dimension.RETURN_REASON_KEY as
    RETURN_REASON_KEY,
    Return_Reason_Dimension.REASON_DESCRIPTION_EN
    as REASON_DESCRIPTION

```

from

```
go_data_warehouse.GOSLDW.dbo.RETURN_REASON_DIMENSION
Return_Reason_Dimension) Return_Reason_Dimension
```

```
on (Return_Reason_Dimension.RETURN_REASON_KEY
    = Returned_Items_Fact.RETURN_REASON_KEY)
```

```
group by
    Time_Dimension.MONTH1,
    Order_Method_Dimension.ORDER_METHOD,
    Return_Reason_Dimension.REASON_DESCRIPTION
```

) D3

```
full outer join
```

```
(select
```

```
    Time_Dimension.MONTH1 as MONTH1,
    Order_Method_Dimension.ORDER_METHOD as
        ORDER_METHOD,
    XSUM(Sales_Fact.QUANTITY for
    Time_Dimension.MONTH1,Order_Method_Dimension.ORDER_M
    ETHOD ) as QUANTITY
```

```
from
```

```
(select
```

```
    Time_Dimension.DAY_KEY as DAY_KEY,
    Time_Dimension.MONTH_EN as MONTH1
```

```
from
```

```
go_data_warehouse.GOSLDW.dbo.TIME_DIMENSION
Time_Dimension) Time_Dimension
```

```
join
```

```
go_data_warehouse.GOSLDW.dbo.SALES_FACT
Sales_Fact
```

```
on (Time_Dimension.DAY_KEY =
    Sales_Fact.ORDER_DAY_KEY)
```



```

join

    (select

        Order_Method_Dimension.ORDER_METHOD_KEY
            as ORDER_METHOD_KEY,
        Order_Method_Dimension.ORDER_METHOD_EN
            as ORDER_METHOD

        from

go_data_warehouse.GOSLDW.dbo.ORDER_METHOD_DIMENSION
Order_Method_Dimension) Order_Method_Dimension

    on (Order_Method_Dimension.ORDER_METHOD_KEY =
        Sales_Fact.ORDER_METHOD_KEY)

group by
    Time_Dimension.MONTH1,
    Order_Method_Dimension.ORDER_METHOD
) D2

on ((D3.MONTH1 = D2.MONTH1) and (D3.ORDER_METHOD =
D2.ORDER_METHOD))

```

Workshop 1: Solution

High-Level Representation of the Overall Query

Select.....from

(Select.....from.....join.....on) D3

Query 1

Full outer join

(Select.....from.....join.....on) D2

Query 2

on ((D3.StitchKey1 = D2.StitchKey1) and
(D3.StitchKey2 = D2.StitchKey2))

select

coalesce(D2.MONTH1,D3.MONTH1) as MONTH1,
coalesce(D2.ORDER_METHOD,D3.ORDER_METHOD) as
ORDER_METHOD,
D3.REASON_DESCRIPTION as REASON_DESCRIPTION,
D2.QUANTITY as QUANTITY,
D3.RETURN_QUANTITY as RETURN_QUANTITY

from

(select

Time_Dimension.MONTH1 as MONTH1,
Order_Method_Dimension.ORDER_METHOD as
ORDER_METHOD,
Return_Reason_Dimension.REASON_DESCRIPTION as
REASON_DESCRIPTION,
XSUM(Returned_Items_Fact.RETURN_QUANTITY for
Time_Dimension.MONTH1,Order_Method_Dimension.ORDER_M
ETHOD,Return_Reason_Dimension.REASON_DESCRIPTION)
as RETURN_QUANTITY

from

(select

Time_Dimension.DAY_KEY as DAY_KEY,
Time_Dimension.MONTH_EN as MONTH1
from

go_data_warehouse.GOSLDW.dbo.TIME_DIMENSION
Time_Dimension)

join

go_data_warehouse.GOSLDW.dbo.RETURNED_ITEMS_FACT
Returned_Items_Fact

on (Time_Dimension.DAY_KEY =
Returned_Items_Fact.DAY_KEY)

join

(select

Order_Method_Dimension.ORDER_METHOD_KEY
as ORDER_METHOD_KEY,
Order_Method_Dimension.ORDER_METHOD_EN
as ORDER_METHOD

from

go_data_warehouse.GOSLDW.dbo.ORDER_METHOD_DIMENSION
Order_Method_Dimension)

on (Order_Method_Dimension.ORDER_METHOD_KEY =
Returned_Items_Fact.ORDER_METHOD_KEY)

join

```
(select
    Return_Reason_Dimension.RETURN_REASON_KEY as
    RETURN_REASON_KEY,
    Return_Reason_Dimension.REASON_DESCRIPTION_EN
    as REASON_DESCRIPTION
from
    go_data_warehouse.GOSLDW.dbo.RETURN_REASON_DIMENSION
    Return_Reason_Dimension)
```

This derived table retrieves rows from the Return Reason Dimension and joins the values with facts from Returned Items Fact on the RETURN_REASON_KEY. The final roll up for RETURN_QUANTITY will also be based on this key as seen in the group by clause. However, Return Reason Dimension is not a conformed dimension and therefore, the same derived table will not be found in the second derived table of this stitch query and it will have no impact on the roll up for the QUANTITY.

```
on (Return_Reason_Dimension.RETURN_REASON_KEY
    = Returned_Items_Fact.RETURN_REASON_KEY)
```

```
group by
    Time_Dimension.MONTH1,
    Order_Method_Dimension.ORDER_METHOD,
    Return_Reason_Dimension.REASON_DESCRIPTION
```

The group by clause, in this case, determines how the RETURN_QUANTITY values will be summed.

) D3

full outer join

D3 is the alias name for the first derived table in this stitch query.

(select

```
Time_Dimension.MONTH1 as MONTH1,
Order_Method_Dimension.ORDER_METHOD as
ORDER_METHOD,
```

The full outer join pertains to the coalesce function which merges the two queries together.

```
XSUM(Sales_Fact.QUANTITY for
Time_Dimension.MONTH1,Order_Method_Dimension.ORDER_M
ETHOD ) as QUANTITY
```

This is the second query in the stitch query. Its projection list requests the conformed dimensions and the other fact, in this case, QUANTITY.

from

```
(select
    Time_Dimension.DAY_KEY as DAY_KEY,
    Time_Dimension.MONTH_EN as MONTH1
from
    go_data_warehouse.GOSLDW.dbo.TIME_DIMENSION
    Time_Dimension)
```

The XSUM indicates that aggregation is occurring for specific groupings.

This derived table retrieves rows from the Time Dimension which are then joined to values from Sales Fact on the DAY_KEY. The final roll up for the report will be done at the month level. The same derived table is also found in the first derived table of this stitch query because Time Dimension is a conformed dimension.

join

```
go_data_warehouse.GOSLDW.dbo.SALES_FACT
Sales_Fact
```

```
on (Time_Dimension.DAY_KEY =
    Sales_Fact.ORDER_DAY_KEY)
```

join

(select

```

Order_Method_Dimension.ORDER_METHOD_KEY
as ORDER_METHOD_KEY,
Order_Method_Dimension.ORDER_METHOD_EN
as ORDER_METHOD

from

go_data_warehouse.GOSLDW.dbo.ORDER_METHOD_DIMENSION
Order_Method_Dimension) Order_Method_Dimension

on (Order_Method_Dimension.ORDER_METHOD_KEY =
Sales_Fact.ORDER_METHOD_KEY)

group by
Time_Dimension.MONTH1,
Order_Method_Dimension.ORDER_METHOD

) D2

on ((D3.MONTH1 = D2.MONTH1) and (D3.ORDER_METHOD =
D2.ORDER_METHOD))

```

This derived table retrieves rows from the Order Method Dimension which are then joined to values from Sales Fact on the ORDER_METHOD_KEY. The final roll up for the report will also be based on this key. The same derived table is found in the first derived table of this stitch query because Order Method Dimension is a conformed dimension.

The group by clause, in this case, determines how the QUANTITY values will be summed. Notice that REASON_DESCRIPTION is not found in this group by clause. That is because it is not a conformed dimension.

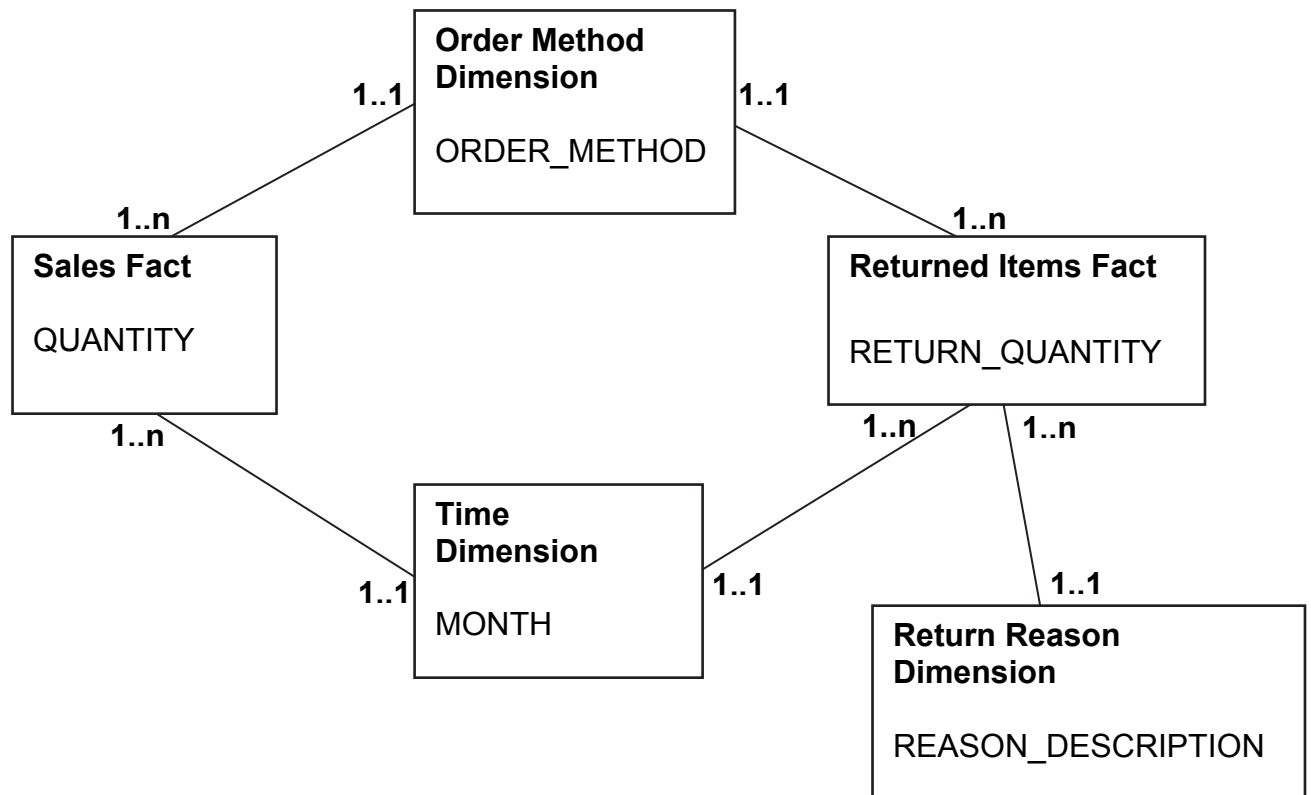
D2 is the alias name for the second derived table in this stitch query.

This portion indicates the join criteria between the two queries of the stitch query. In this case they are joined on columns from the conformed dimensions.

Report Representation Based on Generated SQL

MONTH	ORDER_METHOD	REASON_DESCRIPTION	QUANTITY	RETURN_QUANTITY
MONTH	ORDER METHOD	REASON_DESCRIPTION	QUANTITY	RETURN_QUANTITY
MONTH	ORDER METHOD	REASON_DESCRIPTION	QUANTITY	RETURN_QUANTITY
MONTH	ORDER METHOD	REASON_DESCRIPTION	QUANTITY	RETURN_QUANTITY

Model Diagram Based on Query Subjects Used to Generate SQL



Examine Cognos SQL in Report Studio

- IBM Cognos, by default, summarizes the data.

```

select
    TIME_DIMENSION.MONTH_EN as Month1,
    XSUM(Sales_Fact.Revenue for TIME_DIMENSION.MONTH_EN ) as Revenue,
    XSUM(Sales_Fact.Revenue for TIME_DIMENSION.MONTH_EN ) as
Total_Revenue_
from
    .....
where
    (TIME_DIMENSION.DAY_DATE = Sales_Fact.ORDER_DATE)
group by
    TIME_DIMENSION.MONTH_EN
  
```

Automatic Summarization

Automatic Grouping

Month1 is automatically grouped to roll Revenue and Total Revenue up to each individual instance of a month. This is done at the tabular level of the query. Extra aggregation may be done at the report layout level if summary footers are involved.

If Auto Group & Summarize is turned off in Report Studio, the XSUM does not appear at the tabular level, but will appear at the report level for any summaries in the report layout.

Examine Cognos SQL in Report Studio (cont'd)

- Why do I see two XSUMs?

```
select
    TIME_DIMENSION.MONTH_EN as Month1,
    XSUM(Sales_Fact.Revenue for TIME_DIMENSION.MONTH_EN ) as Revenue,
    XSUM(XSUM(Sales_Fact.Revenue for TIME_DIMENSION.MONTH_EN ) at
    TIME_DIMENSION.MONTH_EN ) as Total_Revenue_
from
    .....
where
    (TIME_DIMENSION.DAY_DATE = Sales_Fact.ORDER_DATE)
group by
    TIME_DIMENSION.MONTH_EN
```

Footer Summarization

Cognos.
software

© 2010 IBM Corporation

When summary footers are created in the report, they are represented by another XSUM as seen in the slide example. Total_Revenue_ is aggregated at the tabular level (Auto Group & Summarize is turned on) as seen by the inner XSUM, and then summarized for a summary footer for a particular grouping in the report layout as seen by the outer XSUM.

Typically, you will see a nested XSUM for footers because auto summarization is enabled in the report. If it were not enabled, there would be no nested XSUM.

If rollup processing is done locally on the IBM Cognos servers, you will see RSUM instead of XSUM as seen below:

```
RSUM(XSUM(Sales_Fact.SALE_TOTAL for Time_Dimension.MONTH1 ) at
Time_Dimension.MONTH1 order by Time_Dimension.MONTH1 asc local)
as SALE_TOTAL1
```


Differentiate Extended vs. Running Aggregates

- Extended aggregate (XSUM, XAVG, XMIN) operations retrieve overall totals.

- Found in generated SQL for queries run in batch mode (PDF, CVS, XML and so on)

XSUM(XSUM(Sales_Fact.Revenue for
TIME_DIMENSION.MONTH_EN) at
TIME_DIMENSION.MONTH_EN) as Total_Revenue_

- Running aggregate (RSUM, RAVG, RMIN) operations calculate totals as they are needed.

- Found in generated SQL for queries run in interactive mode (HTML)

RSUM(XSUM(Sales_Fact.Revenue for
TIME_DIMENSION.MONTH_EN) at
TIME_DIMENSION.MONTH_EN) as Total_Revenue_

If the Rollup Processing property in Framework Manager or Report Studio is set to default, IBM Cognos will generate the appropriate SQL based on the report output type.

Depending on the selected setting, you can force extended aggregation for interactive reports.

For Query Studio, the Framework Manager setting will be used.

Demo 2: Examine Generated SQL in Report Studio

Purpose:

As a modeler, you need to understand SQL patterns that may be generated in Report Studio to help troubleshoot problems encountered by authors, or to simply answer their questions. By testing the model in Report Studio and viewing the generated SQL for different aggregation scenarios, you can see these patterns and learn how to identify what a report is requesting.

Components: Framework Manager, Report Studio

Project: GO Operational

Package: GO Operational (query)


Task 1. Author a simple report in Report Studio and view the generated SQL.

You will first publish the GO Operational (query) package.

1. In the **Project Viewer**, expand **Packages**, and then publish the **GO Operational (query)** package.
2. Launch **IBM Cognos Connection**, log on, and then launch **Report Studio** selecting the **GO Operational (query)** package.
3. Click **Create new**, and then double-click **List**.

4. In the **Insertable Objects** pane, expand **Sales (query)**, and then add the following items to the report:

Query Subject	Query Item
Time	Month
Sales Fact	Revenue

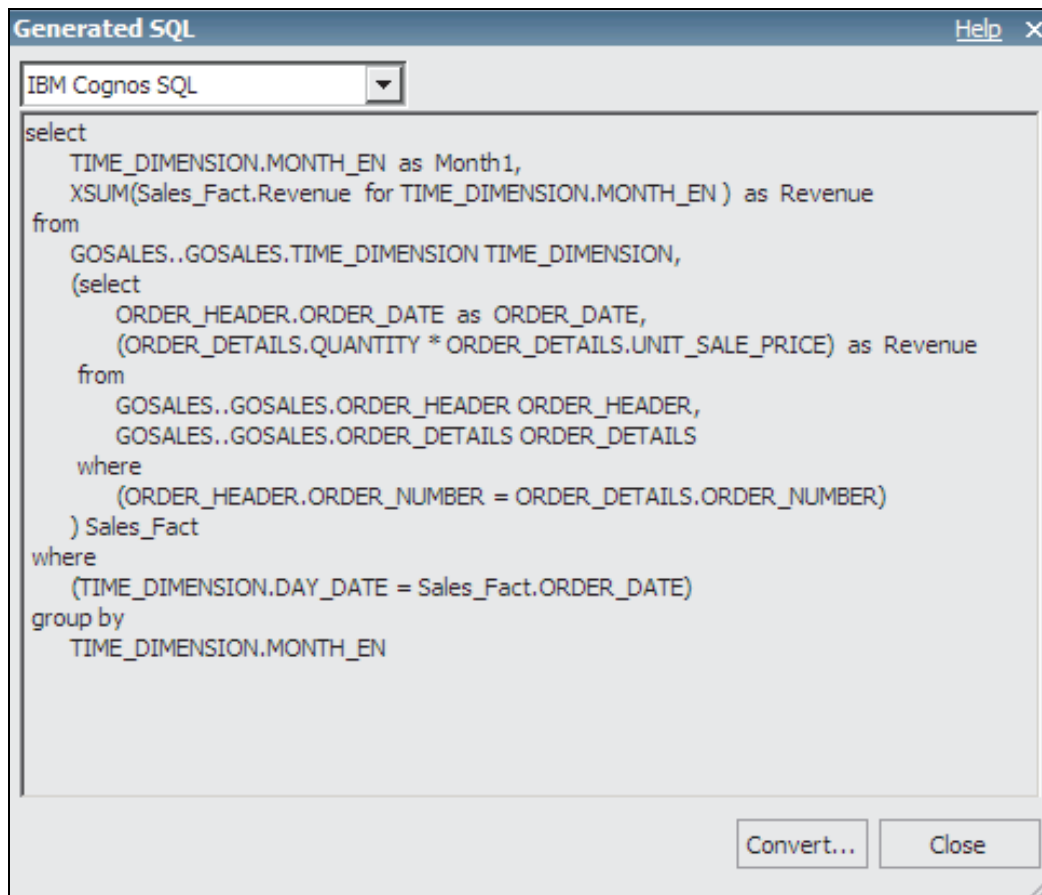
5. To the left of the report, place the cursor over **Query Explorer** , and then click **Query 1**.
6. In the **Properties** pane, click the box beside **Generated SQL**, and then click the **ellipsis**.

A warning message appears indicating that you will be viewing a tabular representation of the query.

Viewing the generated SQL by this method lets you see the SQL specific to this query. Header and footer summaries are not reflected in this SQL because it is specific to the report.

7. Click **OK**, and then from the list, select **IBM Cognos SQL**.



The results appear as follows:



Because the Auto Group & Summarize property is set to Yes by default, you see an XSUM for Revenue. To see detailed rows of data, set this property to No, which tells IBM Cognos not to apply the sum function.

8. Click **Close**.

Task 2. Create a footer and view the generated SQL.

1. Place the cursor over **Page Explorer** , and then click **Page 1**.
2. In the report, click the **Revenue** column header, and then on the **Toolbar** menu, click the **Aggregate**  button, and then click **Total**.

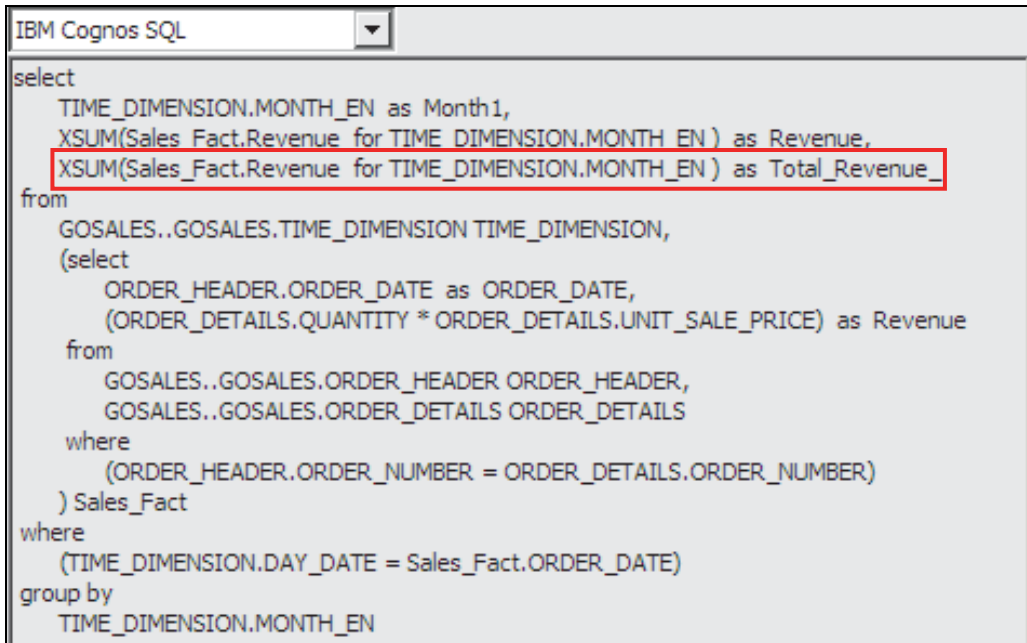
The report appears as shown below:

Month	Revenue
<Month>	<Revenue>
<Month>	<Revenue>
<Month>	<Revenue>
Overall - Total	<Total(Revenue)>

3. Return to the **Query Explorer**, and then click **Query 1**.
4. In the **Properties** pane, view the **Generated SQL**.

- Click **OK**, and then click **IBM Cognos SQL**.

The results appear as follows:



```

select
  TIME_DIMENSION.MONTH_EN as Month1,
  XSUM(Sales_Fact.Revenue for TIME_DIMENSION.MONTH_EN) as Revenue,
  XSUM(Sales_Fact.Revenue for TIME_DIMENSION.MONTH_EN) as Total_Revenue_
from
  GOSALES..GOSALES.TIME_DIMENSION TIME_DIMENSION,
  (select
    ORDER_HEADER.ORDER_DATE as ORDER_DATE,
    (ORDER_DETAILS.QUANTITY * ORDER_DETAILS.UNIT_SALE_PRICE) as Revenue
  from
    GOSALES..GOSALES.ORDER_HEADER ORDER_HEADER,
    GOSALES..GOSALES.ORDER_DETAILS ORDER_DETAILS
  where
    (ORDER_HEADER.ORDER_NUMBER = ORDER_DETAILS.ORDER_NUMBER)
  ) Sales_Fact
where
  (TIME_DIMENSION.DAY_DATE = Sales_Fact.ORDER_DATE)
group by
  TIME_DIMENSION.MONTH_EN
  
```

The SQL appears similar to the previous task except that the tabular SQL is requesting a second summed Revenue as Total Revenue. This does not mean that two identical requests for Revenue will be sent to the database at run time. This just represents the data items that make up the query at the tabular level. The derived table only requests Revenue once. If you look at the native SQL, you will also see only one request is made for revenue aliased as C1.

- Click **Close**, and then from the **Tools** menu, click **Show Generated SQL/MDX**.

Viewing the generated SQL by this method allows you to see the complete SQL statement including footer summary requests.

7. From the list, select **IBM Cognos SQL**.

The results appear as follows:

```

IBM Cognos SQL
select
  TIME_DIMENSION.MONTH_EN as Month1,
  XSUM(Sales_Fact.Revenue for TIME_DIMENSION.MONTH_EN) as
Revenue,
  XSUM(XSUM(Sales_Fact.Revenue for TIME_DIMENSION.MONTH_EN) at
TIME_DIMENSION.MONTH_EN) as Total_Revenue_
from
  GOSALES..GOSALES.TIME_DIMENSION TIME_DIMENSION,
  (select
    ORDER_HEADER.ORDER_DATE as ORDER_DATE,
    (ORDER_DETAILS.QUANTITY * ORDER_DETAILS.UNIT_SALE_PRICE)
as Revenue
  from
    GOSALES..GOSALES.ORDER_HEADER ORDER_HEADER,
    GOSALES..GOSALES.ORDER_DETAILS ORDER_DETAILS
  where
    (ORDER_HEADER.ORDER_NUMBER =
ORDER_DETAILS.ORDER_NUMBER)
  ) Sales_Fact
where
  (TIME_DIMENSION.DAY_DATE = Sales_Fact.ORDER_DATE)
group by
  TIME_DIMENSION.MONTH_EN
  
```

The additional XSUM for the Revenue column populates the report footer.

If you change the Rollup processing property to Local, you will see an RSUM instead of an XSUM. This is illustrated in the Optimize and Tune Framework Manager Models module.

If you turn off Auto Group & Summarize, you will not see nested XSUMs, since you would then be aggregating detailed rows for the footer and not summarized values. You will test this in the next steps.

8. Select **Native SQL**.

You will notice that footer summary information is not requested indicating that the footer summary information will be processed locally on the IBM Cognos servers.

This environment uses a DB2 data source. The request for the footer summary is processed locally because DB2 does not support SQL-OLAP. If the data source were an Oracle database, you would see SQL-OLAP syntax requesting the footer value as shown below:

select

```
"T0"."C0" "MONTH1",      "T0"."C1" "Revenue",
sum("T0"."C1") over ()   "Total_Revenue_"
```

from

```
(.....) "C1"
```

from

```
.....
```

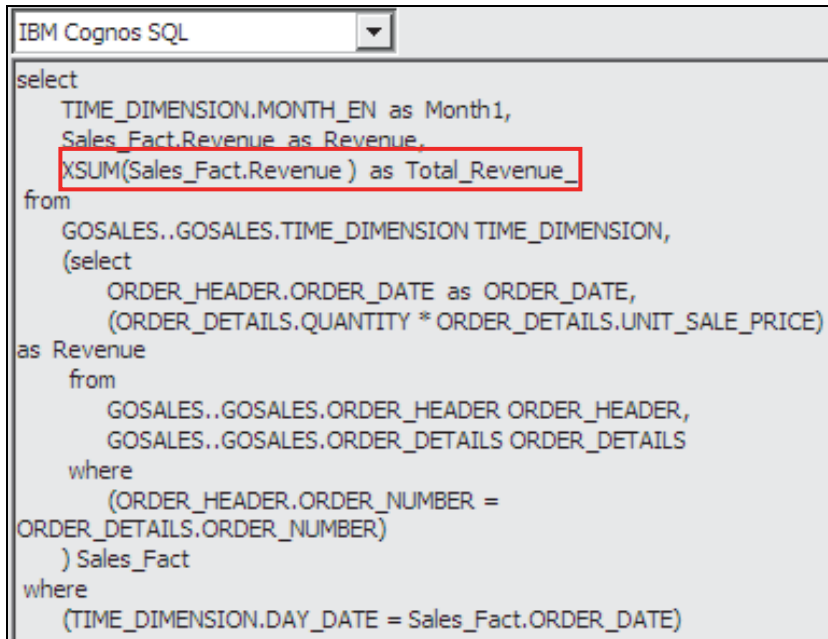
group by

```
.....
```

9. Click **Close**, in the **Properties** pane, click the box beside **Auto Group & Summarize**, and then in the list click **No**.

10. From the **Tools** menu, click **Show Generated SQL/MDX**, and then from the list, click **IBM Cognos SQL**.

The results appear as follows:



```

select
  TIME_DIMENSION.MONTH_EN as Month1,
  Sales_Fact.Revenue as Revenue,
  XSUM(Sales_Fact.Revenue) as Total_Revenue_
from
  GOSALES..GOSALES.TIME_DIMENSION TIME_DIMENSION,
  (select
    ORDER_HEADER.ORDER_DATE as ORDER_DATE,
    (ORDER_DETAILS.QUANTITY * ORDER_DETAILS.UNIT_SALE_PRICE)
as Revenue
  from
    GOSALES..GOSALES.ORDER_HEADER ORDER_HEADER,
    GOSALES..GOSALES.ORDER_DETAILS ORDER_DETAILS
  where
    (ORDER_HEADER.ORDER_NUMBER =
ORDER_DETAILS.ORDER_NUMBER)
) Sales_Fact
where
  (TIME_DIMENSION.DAY_DATE = Sales_Fact.ORDER_DATE)
  
```

Now the footer value, Total_Revenue_, is based on the aggregation of detailed rows rather than summarized values. If you look at the native SQL, you will see that Revenue is now being requested twice from the database. Once as detail rows aliased as C1, and again as a summarized row using the sum function aliased as C0. This type of aggregation is supported by the database and is therefore conducted at the database level.

11. Click **Close**, and then close **Report Studio** without saving the report.
12. Leave IBM Cognos Connection and Framework Manager open for the next demo.

Results:

By testing the model in Report Studio and viewing the generated SQL for different aggregation scenarios, you have seen patterns in the SQL that you can use to identify what a report is requesting.

Examine Generated SQL for Dimensionally Modeled Relational (DMR) Metadata

- Regular dimensions may return un-requested columns in Framework Manager
- Occurs when you test levels that have attributes
- Does not occur in the IBM Cognos studios

When testing regular dimensions in Framework Manager, you may see several columns requested in the generated SQL that you did not select.

This occurs when you test levels that have attributes specified.

This is not the case in the IBM Cognos studios. Only required items for OLAP-style querying are requested.

Demo 3: Examine Generated SQL for Dimensionally Modeled Relational Metadata

Purpose:

When testing DMR levels in either Framework Manager or one of the IBM Cognos studios, you, as a modeler, should be aware of the SQL generation behavior in either scenario so that you are confident in your modeling techniques.

Components: Framework Manager, Report Studio

Project: GO Operational

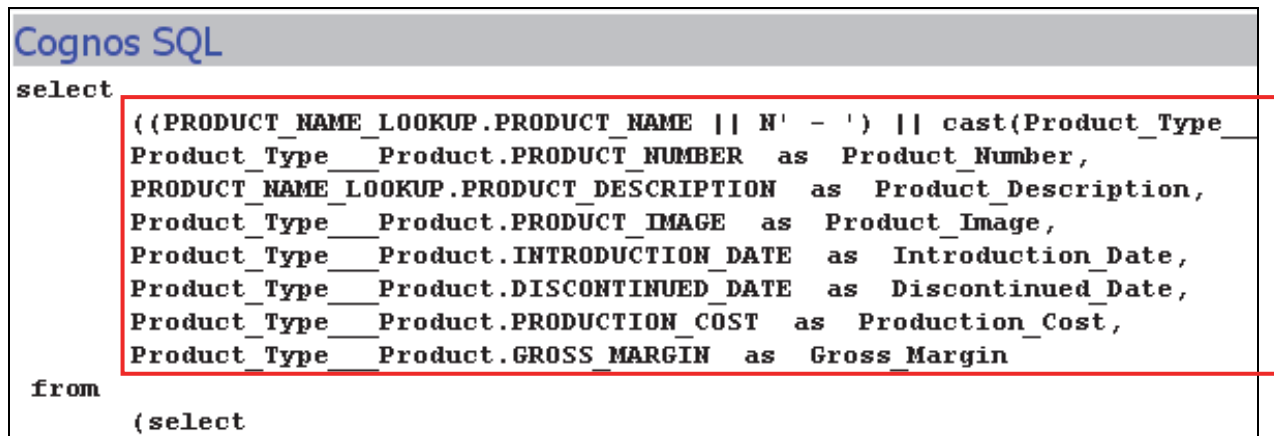
Package: GO Operational (analysis)

Task 1. Test a level in the Products regular dimension in Framework Manager.

1. In Framework Manager, in the **Project Viewer**, expand **Dimensional View>Products>Products**.

2. Test the **Product** level (below Product Type), and then click the **Query Information** tab.

The results appear as follows:



```

Cognos SQL
select
    ((PRODUCT_NAME_LOOKUP.PRODUCT_NAME || N' - ') || cast(Product_Type_
Product_Type__Product.PRODUCT_NUMBER as Product_Number,
PRODUCT_NAME_LOOKUP.PRODUCT_DESCRIPTION as Product_Description,
Product_Type__Product.PRODUCT_IMAGE as Product_Image,
Product_Type__Product.INTRODUCTION_DATE as Introduction_Date,
Product_Type__Product.DISCONTINUED_DATE as Discontinued_Date,
Product_Type__Product.PRODUCTION_COST as Production_Cost,
Product_Type__Product.GROSS_MARGIN as Gross_Margin
from
    (select

```

The projection list is requesting columns other than the business key (PRODUCT_NUMBER) and member caption (PRODUCT_NAME + PRODUCT_NUMBER). PRODUCT_NUMBER, DISCONTINUED_DATE, PRODUCTION_COST, and so on, are all attributes of the Product level and PRODUCT_DESCRIPTION is the member description.





3. Click **Close**, and then double-click the **Products** regular dimension to open the **Dimension Definition** dialog.

4. In the **Hierarchies** pane, click the **Product** level.

The results appear as follows:

Hierarchies:

Products
Product (All)
Product Line
Product Type
Product

 Add Hierarchy
  Add Level
  Delete
  Clear All

☒ Unique Level

Select a level in the hierarchy control to see the query items.

Name	Role	Source
Product Caption	_memberCaption	Products.Product Nam ...
Product Number	_businessKey	Products.Codes.Produ ...
Product Description	_memberDescription	Products.Product Des ...
Product Image		Products.Product Imag ...
Introduction Date		Products.Introduction ...
Discontinued Date		Products.Discontinuec ...
Production Cost		Products.Production C ...
Gross Margin		Products.Gross Margir ...

Here you can see the member description and other attributes that were returned when you tested the Product level. In Framework Manager, these columns are returned to show the contents of the level. These columns would not necessarily be returned in the studios unless requested. Only the member caption and business key are returned with the level's parent keys, which you will see in the next step.

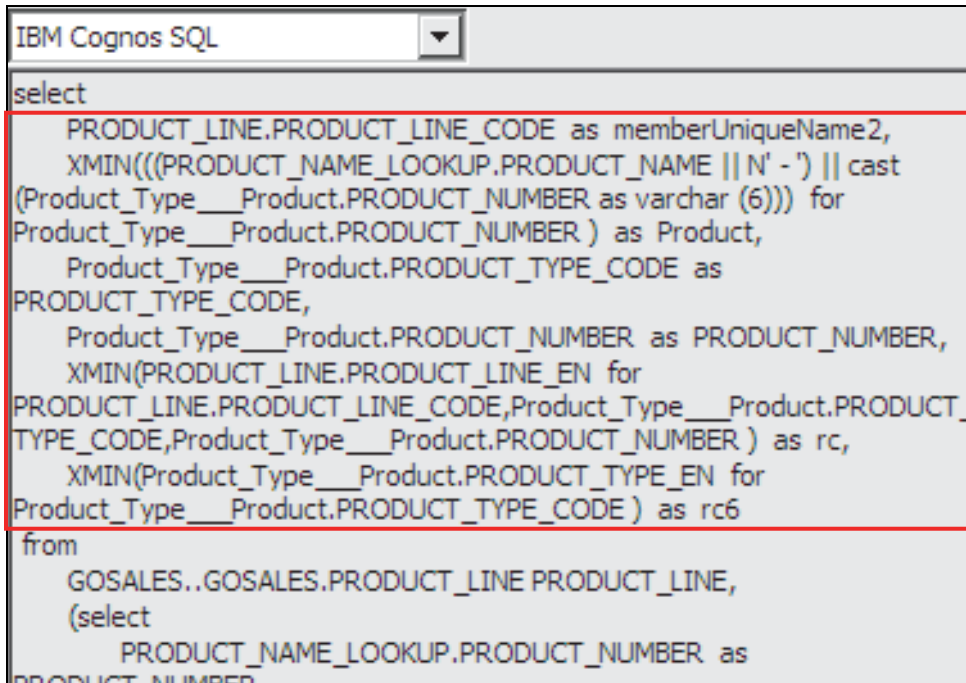
5. Click **Cancel**, and then publish the **GO Operational (analysis)** package.

Task 2. Test a level in the Products regular dimension in Framework Manager.

1. In **IBM Cognos Connection**, launch **Report Studio** selecting the **GO Operational (analysis)** package.
2. Create a new **List** report.
3. In the **Insertable Objects** pane, expand **Sales (analysis)>Products>Products**, and then drag the **Product** level onto the report.
4. From the **Tools** menu, click **Show Generated SQL/MDX**.

- From the list, click **IBM Cognos SQL**.

The results appear as shown below:



```

select
  PRODUCT_LINE.PRODUCT_LINE_CODE as memberUniqueName2,
  XMIN(((PRODUCT_NAME_LOOKUP.PRODUCT_NAME || N' - ' || cast
(Product_Type__Product.PRODUCT_NUMBER as varchar (6))) for
Product_Type__Product.PRODUCT_NUMBER ) as Product,
  Product_Type__Product.PRODUCT_TYPE_CODE as
PRODUCT_TYPE_CODE,
  Product_Type__Product.PRODUCT_NUMBER as PRODUCT_NUMBER,
  XMIN(PRODUCT_LINE.PRODUCT_LINE_EN for
PRODUCT_LINE.PRODUCT_LINE_CODE,Product_Type__Product.PRODUCT
TYPE_CODE,Product_Type__Product.PRODUCT_NUMBER ) as rc,
  XMIN(Product_Type__Product.PRODUCT_TYPE_EN for
Product_Type__Product.PRODUCT_TYPE_CODE ) as rc6
from
  GOSALES..GOSALES.PRODUCT_LINE PRODUCT_LINE,
  (select
    PRODUCT_NAME_LOOKUP.PRODUCT_NUMBER as
PRODUCT_NUMBER

```

The projection list only requests the member caption (PRODUCT_NAME + PRODUCT_NUMBER), the business key (PRODUCT_NUMBER), and parent business key and member caption columns. The parent information is returned to support drill operations such as drill up and down as well as drill-through.

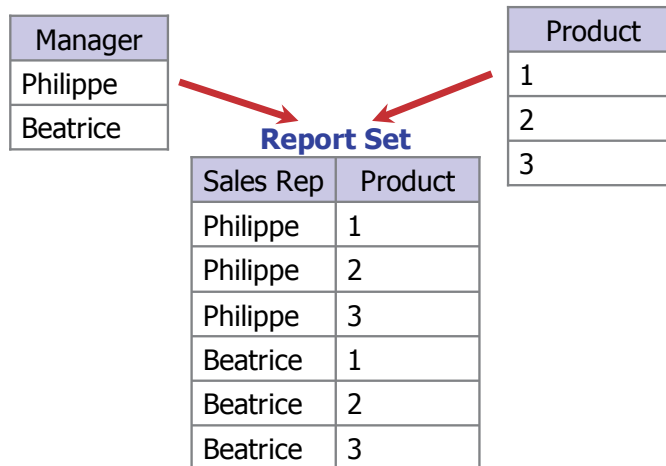
- Click **Close**, and then close all browser windows without saving.
- Close the project without saving, and then close Framework Manager.

Results:

By examining the SQL generation for DMR levels in both Framework Manager and Report Studio, you can see that extraneous columns will not be returned in the studios unless authors request them. Only columns required to support OLAP-style querying will be returned

Define a Cross-Product Join

- A cross-product join combines data by matching each row in one table to each row in another table.
- A cross-product join is not the same as a full outer join.



This slide is a review of the concept of cross-product joins, which you saw earlier in the course.

Cross-product joins are also known as a Cartesian product.

A cross-product join occurs when there is no relationship between two query subjects.

The results and aggregation totals for these types of queries are typically meaningless.

By default, Framework Manager does not allow cross joins because they can be resource intensive.

Identify Cross-Product Join SQL

- Cross-product join SQL simply selects the requested columns from their respective tables.
- No joins are implemented because they do not exist.

```
select
    Manager.Manager as Manager,
    Product.Product as Product
from
    datasource_name.database_name.schema.Manager Manager
    datasource_name.database_name.schema.Product Product
```

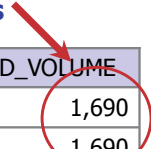
Identify Multi-Fact Query Results

- A contiguous result set uses only conformed dimensions.

MONTH	PRODUCT_NAME	QUANTITY	EXPECTED_VOLUME
April 2004	Aloe Relief	1,410	1,690
April 2004	Bear Edge	574	529
April 2004	Bear Survival Edge	758	954

- In a correlated list, non-conformed dimensions are present.

Repeating values



MONTH	PRODUCT_NAME	ORDER_METHOD	QUANTITY	EXPECTED_VOLUME
April 2004	Aloe Relief	Telephone	286	1,690
April 2004	Aloe Relief	Web	854	1,690
April 2004	Aloe Relief	E-mail	270	1,690
April 2004	Bear Edge	Telephone	224	529
April 2004	Bear Edge	Web	60	529

Interpreting a multi-fact query is not as easy as authoring one. It is important to understand the impact of conformed and non-conformed dimensions on a multi-fact query, and the level of granularity and additive nature of the data.

A contiguous result set means the results of each fact query can be mapped to each other with 0..1 to 1..0 precision.

A correlated list refers to a looser coupling of the data in which non-conformed dimensions have been introduced.

Identify Contiguous Result Sets

- Multi-fact queries with only conformed dimensions can have different levels of granularity.

Common Grain from Conformed Dimensions

MONTH	PRODUCT_NAME	QUANTITY	EXPECTED_VOLUME
April 2004	Aloe Relief	1,410	1,690
April 2004	Bear Edge	574	529
April 2004	Bear Survival Edge	758	954

Repeating
Values

Different Levels of Granularity from Conformed Dimension

MONTH	DAY_DATE	PRODUCT_NAME	QUANTITY	EXPECTED_VOLUME
April 2004	Apr 25, 2004	Aloe Relief	286	1,690
April 2004	Apr 27, 2004	Aloe Relief	854	1,690
April 2004	Apr 28, 2004	Aloe Relief	270	1,690
April 2004	Apr 2, 2004	Bear Edge	224	529
April 2004	Apr 4, 2004	Bear Edge	60	529
April 2004	Apr 7, 2004	Bear Edge	166	954

Cognos.
software

© 2010 IBM Corporation

Identify Contiguous Result Sets (cont'd)

- Adding a lower level than the common level of granularity between the two facts, presents repeating values for the fact at the higher level.
- Determinants will prevent double-counting.

Multi-Fact/Multi-Grain Report

MONTH	DAY_DATE	PRODUCT_NAME	QUANTITY	EXPECTED_VOLUME
April 2004	Apr 25, 2004	Aloe Relief	286	1,690
April 2004	Apr 27, 2004	Aloe Relief	854	1,690
April 2004	Apr 28, 2004	Aloe Relief	270	1,690
April 2004	Apr 2, 2004	Bear Edge	224	529
April 2004	Apr 4, 2004	Bear Edge	60	529
April 2004	Apr 7, 2004	Bear Edge	166	954
Summary			2,215,354	2,166,005

With determinants specified, IBM Cognos will not double-count the values at the higher level of granularity.

As seen in the slide example, EXPECTED_VOLUME is not double counted because the determinants specified on the Time Dimension dictate that those values should be aggregated to, and grouped by, the month key.

Troubleshoot Unexpected Results

- When the data results are unexpected, check the SQL.
 - What kind of joins do you see?
 - Do they correspond with the relationships you have in Framework Manager?
 - Do you see a stitch query that you do not expect?
 - Do you see a stitch query without all required conformed dimensions?
 - Do you see a stitch query with no conformed dimensions?
- Do you have determinants specified for multiple levels of granularity in a single query subject?
- Are you using the correct SQL options?
- Check the native SQL to see if it is what you expected.

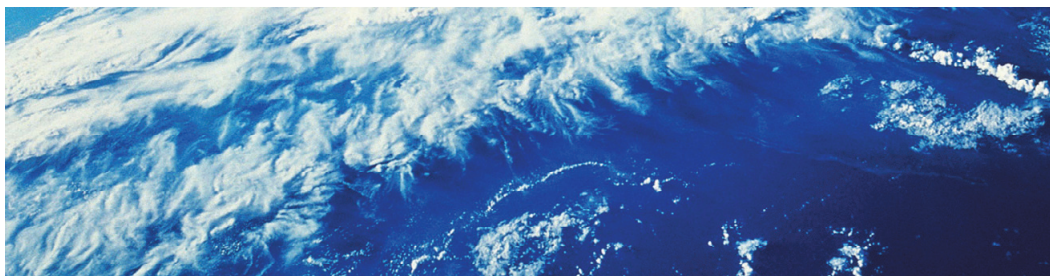
Summary

- You should now be able to identify:
 - governors that affect SQL generation
 - stitch query SQL
 - conformed and non-conformed dimensions in generated SQL
 - multi-fact/multi-grain stitch query SQL
 - variances in Report Studio generated SQL
 - dimensionally modeled relational SQL generation
 - cross join SQL
 - various results sets for multi-fact queries



Use Advanced Parameterization Techniques in Framework Manager

IBM Cognos BI



Business Analytics

© 2010 IBM Corporation

Objectives

- At the end of this module, you should be able to:
 - identify session and model parameters
 - leverage session, model, and custom parameters
 - create prompt macros
 - leverage macro functions associated with security

IBM Cognos Session Parameters

- A variable associated with a IBM Cognos session
- There are two types of session parameters:
 - environment
 - model

Parameter	Value	Override Value
account.defaultName	Admin Person	
account.personalInfo.email	admin@grtd123.com	
account.personalInfo.givenName	Admin	
account.personalInfo.surname	Person	
account.personalInfo.userName	admin	
current_timestamp	2008-10-08 11:04:25....	
machine	TP-KAMALA	
runLocale	en	

Cognos.
software

© 2010 IBM Corporation

Each session parameter must have a name and a default value to ensure the session parameter resolves to a value at run time.

You cannot have more than one session parameter with the same name.

Override values can be set for testing inside the model only.

Business Analytics



Review Environment Session Parameters

- Predefined and stored in the content store database

Parameter	Value	Override Value
account.defaultName	Admin Person	
account.personalInfo.email	admin@grtd123.com	
account.personalInfo.givenName	Admin	
account.personalInfo.surname	Person	
account.personalInfo.userName	admin	
current_timestamp	2008-10-13 12:17:27.014-05:00	
machine	TP-KAMALA	
runLocale	en	

Cognos.
software

© 2010 IBM Corporation

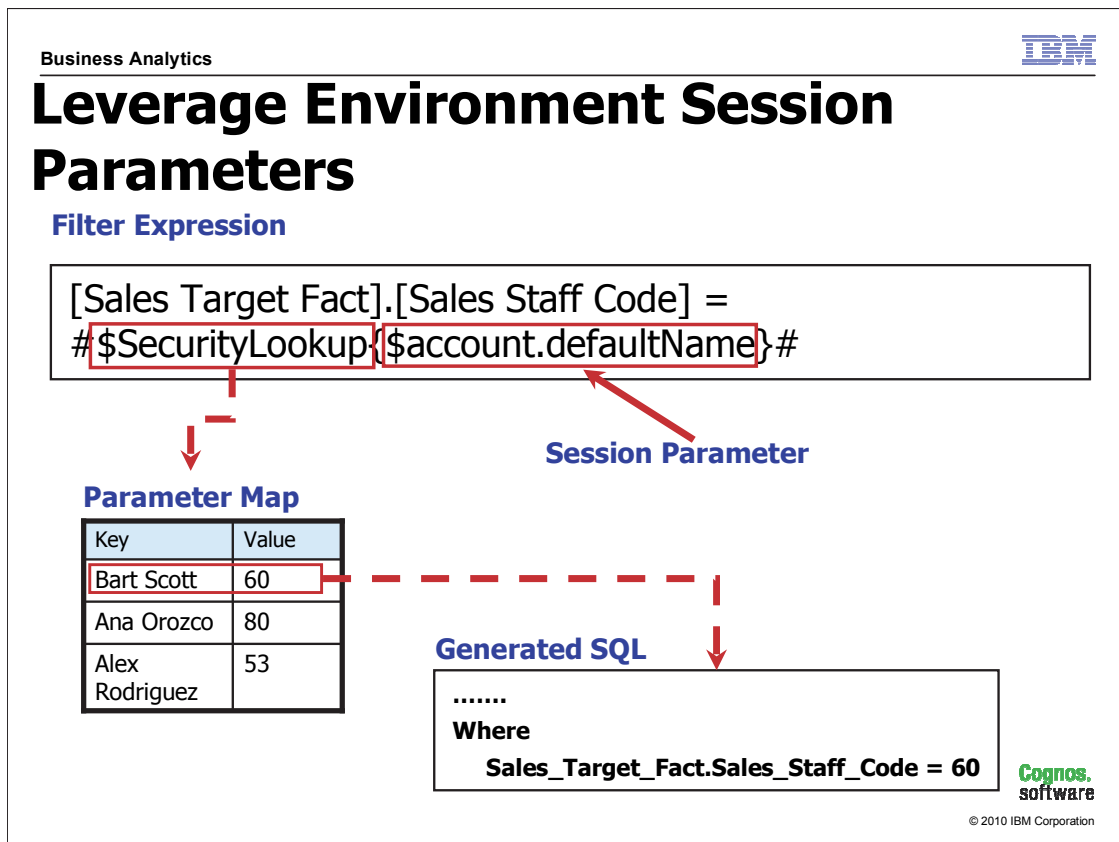
This topic was briefly discussed in the Calculations and Filters module. Here are some more details.

Depending on the authentication provider, you may see more or fewer session parameters.

By default, the following session parameters appear in Framework Manager:

- account.defaultName
- account.personalInfo.userName
- runLocale

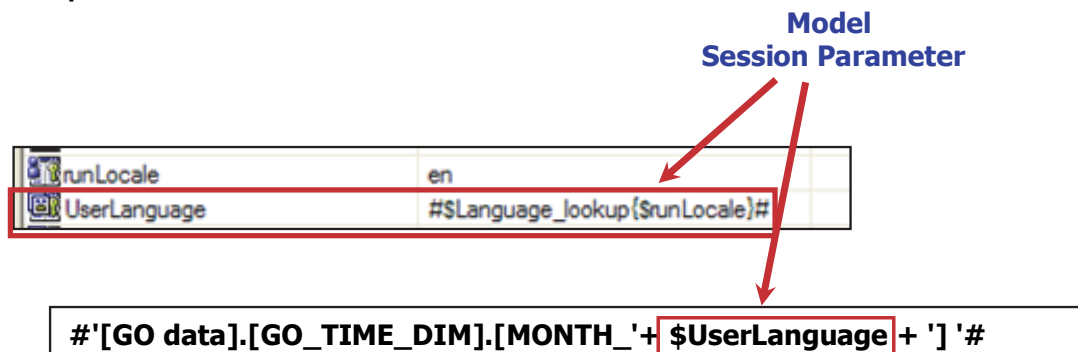
If you log on anonymously, you will see only runLocale and account.defaultName(Anonymous).



As seen earlier in the course, use environment session parameters to create a dynamic model by using them in filters, SQL, property settings, and other model objects.

Model Session Parameters

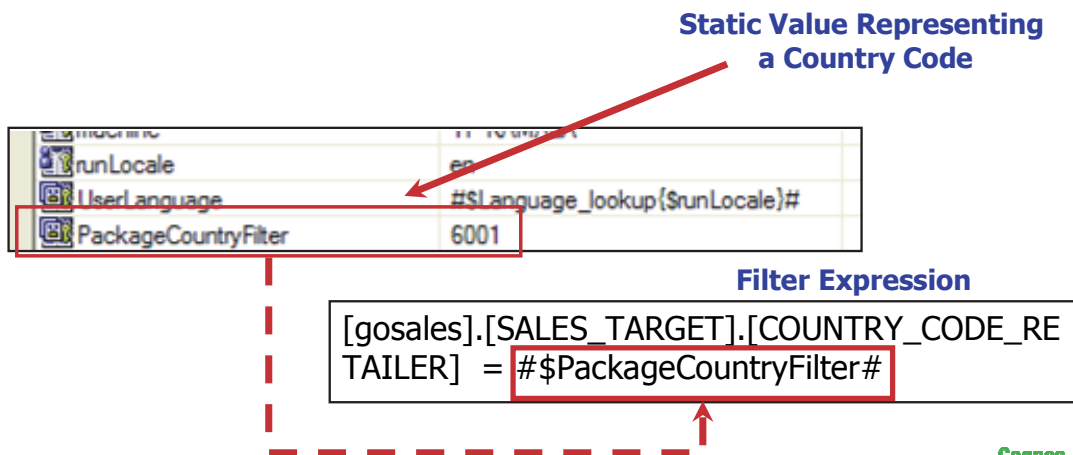
- Created in Framework Manager and published with every package
- Can include the use of existing environment session parameters and static values



Model session parameters can be useful when you are trying to centralize the maintenance on a macro that is used throughout a model.

Filter Data Based on Package Access

- Create model session parameters with static values
- Values are published with the package



By using a model session parameter, you can centrally control a value throughout your model. In this case the value is static and controls which country for which a user can see values.

The filter expression in the slide example generates a Where clause in which Country code equates to 6001, which is the value specified in the model session parameter.

Demo 1: Leverage Model Session Parameters

Purpose:

You have been asked to centralize the language lookup macro to save time when modeling future imports of new tables or data sources. You will do this by placing the macro in a model session parameter and using it throughout the model.

You have also been asked to let report authors automatically filter data based on which package they choose. They would like to filter data based on countries. You will use a model session parameter and a model filter to fulfill this request.

Components: Framework Manager, Business Insight Advanced

Project: GO Operational










Packages: GO Operational (query) - US, GO Operational (query) - France

Task 1. Centralize the language lookup macro.

1. In **Framework Manager**, close any projects that may be open, and then open the **GO Operational** project located at **C:\Edcognos\B5152\CBIFM-Start Files\Module 19\GO Operational**.
2. If prompted, log in as User ID **admin**, and Password **Education1!**.
3. From the **Project** menu, click **Session Parameters**.
4. Click **New Key**, and then in the highlighted **Parameter** field, type **UserLanguage**.

5. In the **Value** field, type **#\$Language_lookup{\$runLocale}#**

The results appear as follows:

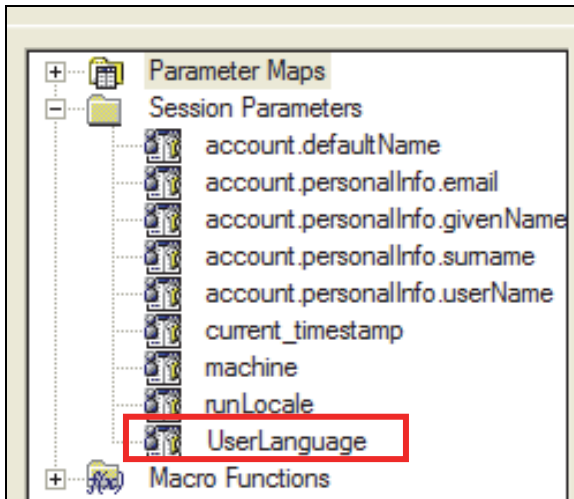
Parameter	Value	Override Value
 account.defaultName	Admin Person	
 account.personalInfo.email	admin@grtd123.com	
 account.personalInfo.givenName	Admin	
 account.personalInfo.surname	Person	
 account.personalInfo.userName	admin	
 current_timestamp	2008-10-22 16:08:58.671-05:00	
 machine	CLASSBASER	
 runLocale	en	
 UserLanguage	#\$Language_lookup{\$runLocale}#	

Note: For NTLM, if an email address is not specified in the user's personal information in IBM Cognos Connection, you will not see an email entry in the session parameters.

6. Click **OK**.
7. In the **Project Viewer**, expand **GO Operational Model>Consolidation View>Order Method**, and then double-click **Order Method**.

8. Under **Available Components**, click the **Parameters** tab, and then expand **Session Parameters**.

The results appear as follows:



Notice the UserLanguage parameter. This is the new model session parameter you just created.

9. In the expression, replace **\$Language_lookup{\$runLocale}** with the **UserLanguage** parameter.

The results appear as follows:

```
#'[gosales].[ORDER_METHOD].[ORDER_METHOD_' +
$UserLanguage + ']'#
```

10. Click the **Test Sample** button  to ensure the macro works, and then click **OK**.

11. Apply the same technique to **Product Line** and **Product Type** in the **Products** query subject.

The results for each expression appear as follows:

```
#'[gosales].[PRODUCT_LINE].[PRODUCT_LINE_' + $UserLanguage + ']'#
```

```
#'[gosales].[Product Type & Product].[PRODUCT_TYPE_' +  
$UserLanguage + ']'#
```

If time permits, you can implement this technique for all language-based calculations in the project.











12. Save the project as **C:\Edcognos\B5152\Course_Project\Params\GO Operational.cpf**.

Task 2. Create a model session parameter to filter packages.

1. From the **Project** menu, click **Session Parameters**.
2. Click **New Key**, and then in the highlighted **Parameter** field, type **PackageCountryFilter**.
3. In the **Value** field, type **1003**.

This value represents the country code for the United States.

The Session Parameters dialog box appears as shown below:

Parameter	Value	Override Value
 account.defaultName	Admin Person	
 account.personallInfo.email	admin@grtd123.com	
 account.personallInfo.givenName	Admin	
 account.personallInfo.surname	Person	
 account.personallInfo.userName	admin	
 current_timestamp	2008-10-22 16:23:54.062-05:00	
 machine	CLASSBASER	
 runLocale	en	
 UserLanguage	#\$Language_lookup{\$runLocale}#	
 PackageCountryFilter	1003	

4. Click **OK**.

Task 3. Apply a country filter using a model session parameter.

1. In the **Project Viewer**, in the **Consolidation View** namespace, double-click **Sales Target Fact**.
2. Click the **Filters** tab, then click **Add**.
3. In the **Name** box, type **Country Filter**.
4. From **Foundation Objects View**>**gosales**>**SALES_TARGET**, add **COUNTRY_CODE_RETAILER** to the expression, followed by **=**.
5. Under **Available Components**, click the **Parameters** tab, and then expand **Session Parameters**.
6. Double-click **PackageCountryFilter** to add it to the expression definition.

The results appear as follows:

```
[gosales].[SALES_TARGET].[COUNTRY_CODE_RETAILER] =  
#$$PackageCountryFilter#
```

Of course, you would apply similar filters on all fact query subjects published in the packages to ensure the model is truly region specific. For the purposes of this demonstration, you will apply the filter to only one fact query subject.

7. Click **OK**, and then click **OK** again.

8. Test the following items in the **Consolidation View** with **Auto Sum** enabled:

Query Subject	Query Item
Sales Target by Location	Sales Target Country
Sales Target Fact	Sales Target

The results appear as follows:

Test results	
Sales Target Country	Sales Target
United States	742360700

Only a value for the United States is returned.

9. Click **Close**, and then save the project.

Task 4. Create and publish a filter-based package.

You will create a package that will be filtered on the country of the United States.

1. In the **Project Viewer**, expand **Packages**, copy **GO Operational (query)**, paste the copy under **Packages**, and then rename the copy to **GO Operational (query) - US**.
2. Publish the **GO Operational (query) - US** package.

This package has now been published with a hard-coded model session parameter of 1003, which is the country code for the United States.

Task 5. Create and publish a second filter-based package.

You will now create a second package that will be filtered on the country of France.

1. Create another copy of **GO Operational (query)**, and then rename it to **GO Operational (query) - France**.

Before you can publish this package, you must change the PackageCountryFilter model session parameter value to represent the country code for France.

2. From the **Project** menu, click **Session Parameters**, change the value for **PackageCountryFilter** from **1003** to **6001**, and then click **OK**.
3. Publish the **GO Operational (query) - France** package.

When you publish filter-based packages, make sure you have the correct model session parameter value set. For example, before publishing GO Operational (query) - France, set the PackageCountryFilter value to 6001. Change it back to 1003 before publishing GO Operational (query) - US.

4. Save your project.

Task 6. Test filter-based packages in Business Insight Advanced.

1. Launch **IBM Cognos Connection**, log on, and then launch **Business Insight Advanced** selecting the **GO Operational (query) - France** package for a **List** report.
2. Expand **Sales Targets (query)**, and then add the following items to the report:

Query Subject	Query Item
Sales Target by Location	Sales Target Country
Sales Target Fact	Sales Target

Only data for France appears.

Sales Target Country	Sales Target
France	257,675,400
Overall - Summary	257,675,400

3. Return to **IBM Cognos Connection**, and do not save the report.
4. Launch **Business Insight Advanced** again selecting the **GO Operational (query) - US** package.
5. Create the same report.

This time only data for the United States appears.

Sales Target Country	Sales Target
United States	742,360,700
Overall - Summary	742,360,700

6. Return to **IBM Cognos Connection**, and do not save the report, and leave Framework Manager open for the next demo.

Results:

You centralized a widely used macro and created filter-based packages that let report authors automatically filter data based on the package they selected.


Business Analytics



Custom Environment Session Parameters

- Requires a customizable LDAP security provider
- Are exposed to IBM Cognos through IBM Cognos Configuration
- Appear as session parameters in Framework Manager

**Custom
Session
Parameter**



Parameter	Value	Override Value
account.defaultName	Frank Bretton	
account.parameters.Location	Calgary	
account.personalInfo.businessPhone	1 (403) 232-5986	
account.personalInfo.email	FBretton@grtd123.com	
account.personalInfo.faxPhone	1 (403) 232-5831	
account.personalInfo.givenName	Frank	
account.personalInfo.surname	Bretton	
account.personalInfo.userName	brettonf	
current_timestamp	2008-10-13 16:49:22.267-05:00	

Cognos.
software

© 2010 IBM Corporation

Microsoft Active Directory and Sun Java System Server are both supported with respect to custom environment session parameters.

Attributes available from an LDAP security provider can be exposed to IBM Cognos as custom properties through IBM Cognos Configuration.

Demo 2: Leverage Custom Environment Session Parameters

Purpose:


Report authors have requested the ability to filter their reports based on their location. To fulfill this request, you will leverage a custom property in The Great Outdoors Company LDAP authentication provider. You will make the custom property available as a session parameter in Framework Manager and then use that parameter in a model filter.

Components: IBM Cognos Configuration, Framework Manager, Business Insight Advanced


Project: Go Operational

Package: GO Operational (query)

Task 1. Configure IBM Cognos to use a custom session parameter.

1. From the **Start** menu, point to **All Programs>IBM Cognos 10**, and then click **IBM Cognos Configuration**.
2. In the **Explorer** pane, under **Security>Authentication**, click **LDAP**.
3. In the **Properties** pane, scroll down to the **Custom properties** row, click inside the **Value** column, and then click **Edit** .
4. In the **Value** dialog box, click **Add**, in the **Name** field, type **Location**, and then in the **Value** field, type **l** (lower case L).

This value is associated with the custom parameter for locality in the LDAP authentication provider.

- Click **OK**, save the configuration, and then click **Restart**  to restart the IBM Cognos service.

Note: If you receive a warning message about the mail server connection, disregard it. A mail server is not used in this course.

- Close **IBM Cognos Configuration**.

Task 2. View the custom session parameter in Framework Manager.

- In **Framework Manager**, from the **Project** menu, click **Logoff**.
- Click **Log on again**, and then log on as **brettonf** (password=**Education1!**).
Frank Bretton is a report author that works out of the Calgary office.
- From the Project menu, click **Session Parameters**.

The results appear as follows:

Parameter	Value	Override Value
account.defaultName	Frank Bretton	
account.parameters.Location	Calgary	
account.personalInfo.businessPhone	1 (403) 232-5986	
account.personalInfo.email	FBretton@grtd123.com	
account.personalInfo.faxPhone	1 (403) 232-5831	
account.personalInfo.givenName	Frank	
account.personalInfo.surname	Bretton	
account.personalInfo.userName	brettonf	
current_timestamp	2008-10-13 16:49:22 -05:00	

For the logged on user, there is an account.parameters.Location parameter available with a value of Calgary, the city where this employee works.

- Click **Cancel**.

Task 3. Create a model filter that uses the custom parameter.

1. In the **Project Viewer**, under **Presentation View**, right-click the **GO Operational Sales (query)** namespace, point to **Create**, and then click **Filter**.
2. In the **Name** box, type **My Location**.
3. In the **Available Components** pane, expand **Consolidation View> Staff by Location**.
4. Create the following expression:

[Consolidation View].[Staff by Location].[Staff City] =

5. Under **Available Components**, click the **Parameters** tab, expand **Macro Functions**, and then double-click **sq** to add it to the expression definition.

Tip: You can view syntax assistance for macro functions by selecting them and viewing the Tips pane. There you can see explanations of the functions as well as examples.

6. Expand **Session Parameters**, and then double-click **account.parameters.Location** to add it to the expression definition.

The expression appears as shown below:

**[Consolidation View].[Staff by Location].[Staff City] =
#sq(\$account.parameters.Location)#**

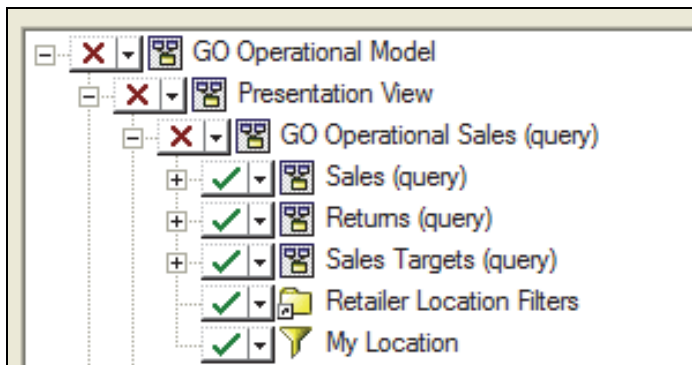
This filter uses the custom environment session parameter called Location to focus a query.

7. Click **OK**, and then save the project.

Task 4. Test model filter in Business Insight Advanced.

1. In the **Project Viewer**, under **Packages**, double-click **GO Operational (query)**, and then select **My Location** to add it to the package.

The results appear as follows:



2. Click **OK**, and then publish the **GO Operational (query)** package.
3. In **IBM Cognos Connection**, log on as **brettonf** (password = **Education1!**).
4. Launch **Business Insight Advanced** selecting the **GO Operational (query)** package for a **List** report.
5. In the **Insertable Objects** pane, expand **Sales (query)**, and then add the following items to the report:

Query Subject	Query Item
Staff by Location	Staff City
Sales Fact	Revenue

All cities and their revenue appear.

6. From the **Insertable Objects** pane, drag the **My Location** filter to the report.
7. Click **OK**.

The results appear as follows:

Staff City	Revenue
Calgary	\$111,146,739.19
Overall - Summary	\$111,146,739.19

Because you are logged in as Frank Bretton, the report is filtered on his city, Calgary, using the custom environment session parameter.

8. Return to **IBM Cognos Connection** without saving the report.

Results:

By leveraging a custom property in The Great Outdoors Company LDAP authentication provider, you let report authors filter their reports based on their location.

Examine the Scope of Session Parameters

- Session parameters can be used in:
 - object properties
 - SQL
 - filters and calculations
 - stored procedure arguments
 - model session parameter value

Session parameters can be used in macros to dynamically change many model elements, including:

- object properties such as Content manager datasource, Catalog, and Schema properties in data source connections
- SQL in a data source query subject
- filter and calculation syntax
- and so on

Demo 3: Dynamically Change Your Data Source Connection

Purpose:

The Great Outdoors Company has different databases with the same structure but different data to serve different regions of the business. You will allow dynamic selection of a database based on the current user by using a macro and session parameter to change the Content Manager Datasource connection name at run time.

Components: Framework Manager, IBM Cognos Connection

Project: GO Operational

Task 1. Create a data source lookup parameter map.

1. In **Framework Manager**, in the **GO Operational** project, right-click **Parameter Maps**, point to **Create** and then click **Parameter Map**.
2. In the Name box, type **DataSourceLookup**, and then click **Next**.
3. Under **Key**, in the first row, type **brettonf** and then under **Value** type **GOSALES**.
4. Under **Key**, in the second row, type **scottb**, and then under **Value**, type **GOSALES2**.

The results appear as follows:

Key	Value
brettonf	GOSALES
scottb	GOSALES2

This parameter map will map the \$account.personalInfo.userName value to the appropriate data source.

5. Set **Default value** to **GOSALES**, and then click **Finish**.

Task 2. Create a second data source connection in IBM Cognos Connection.

You will now create a second data source connection to the GOSALES database to mimic the ability to dynamically switch from one database to another depending on the current user.

1. Log on to **IBM Cognos Connection** as **admin** (password = **Education1!**).
2. Click **Administer IBM Cognos content**, and then click the **Configuration** tab.
3. Create a new data source using the following parameters:
 - Name: **GOSALES2**
 - Type: **IBM DB2**
 - Database name: **GS_DB**
 - Select **Password**
 - User ID: **GOSALES**
 - Password: **Education1!**
 - Confirm Password: **Education1!**
4. Click **Finish**.

Task 3. Add a macro to the data source properties.

1. In **Framework Manager**, select the **GOSALES** data source.
2. In the **Properties** pane, click in the **Content Manager Datasource** field, and then click the **ellipsis**.
3. Delete the **GOSALES** text, and then click the **Insert Macro** button.
4. Under **Available components**, expand **Parameter Maps**, and then double-click **DataSourceLookup** to add it to the definition.

- Expand **Session Parameters**, and then drag **account.personalInfo.userName** into the brackets of the **DataSourceLookup** parameter map.

The results appear as follows:

```
#$DataSourceLookup{$account.personalInfo.userName}#
```

- Click **OK**.

The Value pane now displays the new macro.

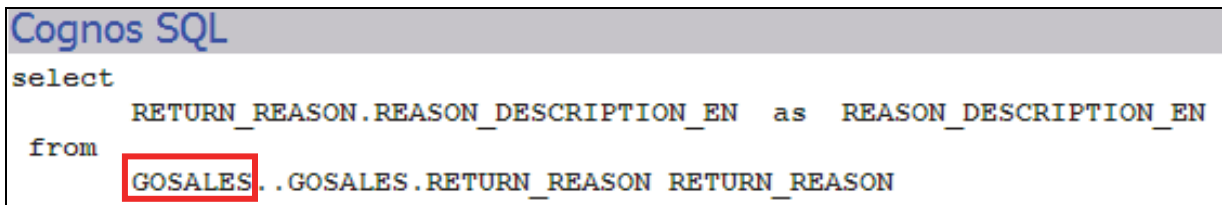
- Click **OK**.

Task 4. Test the data source property macro.

You are currently logged on as Frank Bretton. You will test this account first.

- In the **Project Viewer**, under **Consolidation View>Return Reason**, test the **Return Reason Description** query item (you can choose to test any other query item if you wish).
- Click the **Query Information** tab.

The results appear as follows:



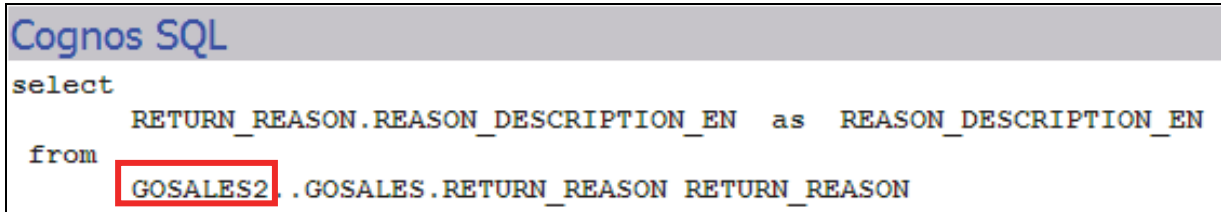
```
Cognos SQL
select
    RETURN_REASON.REASON_DESCRIPTION_EN as REASON_DESCRIPTION_EN
from
    GOSALES.GOSALES.RETURN_REASON RETURN_REASON
```

Notice the data source connection name. You will now log on as a different user to dynamically use another data source connection.

- Click **Close**.
- Log off, and then log on again with **scottb** (password = **Education1!**).

5. Test the query item again.

The results appear as follows:



```

Cognos SQL
select
    RETURN_REASON.REASON_DESCRIPTION_EN as REASON_DESCRIPTION_EN
from
    GOSALES2.GOSALES.RETURN_REASON RETURN_REASON
  
```

Notice the data source connection name has changed.

For this technique to work, ensure that the structure of the underlying data sources match. In other words, the database structures must be mirror images with the exception of the data they contain.

6. Click **Close**, log off, log on as **admin** (password = **Education1!**), and then save the project.

If you test the query item again, the default value GOSALES in the parameter map would be used since the admin user is not a value in the parameter map.

Note: You can achieve the same effect in IBM Cognos Connection by configuring your data source connections appropriately. For example, you can create one Data Source in IBM Cognos Connection, and then multiple connections for that data source. Each connection would point to a different version of the database containing region specific data. You would then apply security on each of the connections. When a user accesses the data source, they will automatically use the connection to which they have access. If a user has access to more than one connection, they will be prompted at run time to select the connection they want to use. To support multiple connections for a data source in your Framework Manager project, you must clear the data source catalog and schema properties. This information will be obtained from the connection information at run time.

Results:

Using a macro and a session parameter, you were able to dynamically change your data source connection based on the current user.

Examine Prompt Macros

- Prompt macros allow sophisticated prompting with the ability to:
 - provide prompt names
 - specify data types
 - provide default values
 - use Prompt Info properties
 - append text
 - make filters and prompts optional
 - request single or multiple values

Data types include the token data type, which is used for passing SQL syntax.


You can reference query items allowing the use of Prompt Info properties setting.

You can append text at the beginning or end of the prompt.

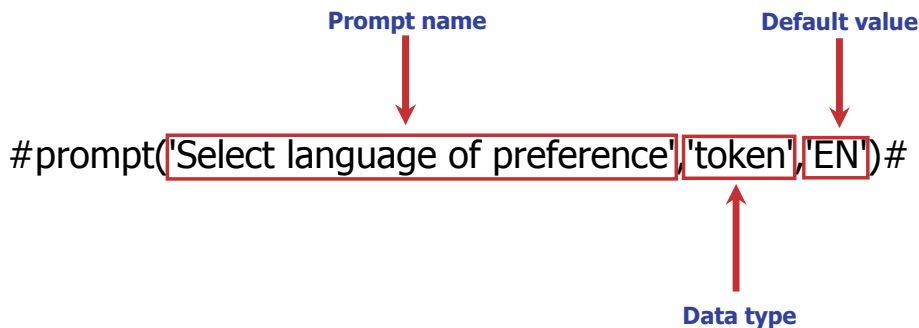
Specify a default value to make the prompt optional. If no value is specified the prompt is mandatory.

You can use the prompt function for single values or the promptmany function for multiple values.

Business Analytics



Use the Prompt Macro Function




The diagram illustrates the syntax of the prompt macro function: `#prompt('Select language of preference','token','EN')#`. Three red arrows point to specific parts of the macro: one from the label 'Prompt name' to the first single-quoted string, one from 'Default value' to the third single-quoted string, and one from 'Data type' to the second single-quoted string.

Prompt name

Default value

Data type

`#prompt('Select language of preference','token','EN')#`


© 2010 IBM Corporation

The prompt macro above allows the end user to enter a single value, in this case a token value, to dynamically affect the SQL generated for selecting a language based column.

Use Preceding Text in a Prompt

The diagram shows a prompt macro: `# prompt('Enter data source number' 'int' 'great_outdoors_warehouse', 'great_outdoors_warehouse')#`. Annotations with red arrows point to specific parts: 'Prompt name' points to the first single quote, 'Data type' points to the second single quote, 'Default value' points to the first instance of 'great_outdoors_warehouse', and 'Preceding text' points to the second instance of 'great_outdoors_warehouse'.

```
# prompt('Enter data source number' 'int' 'great_outdoors_warehouse', 'great_outdoors_warehouse')#
```

This prompt macro uses the prompt function with preceding text to allow end users to select the data source to which they would like to connect.

The prompt macro uses preceding text to specify the data source name which is concatenated to a value supplied by the user. If the user supplies a value of 2, it will result in a data source name of GOSALES2.

Demo 4: Create Prompt Macros to Filter Data

Purpose:

Report authors would like the ability to specify which language they view the data in at run time. This applies to all queries they write. To facilitate this, you will alter your centralized language macro to generate a prompt at run time.

Rather than connect to a data source based on their user ID, authors have requested to choose the data source to which they would like to connect. You will use a prompt function in the data source properties to accomplish this.

Authors would also like to be prompted for the return reason for returned items and be able to supply one or more values for Return reason code.

Component: **Framework Manager**

Project: **GO Operational**

Task 1. Create a prompt macro to select a specific language column based on user input.

The following macro will use the token data type to pass SQL at run-time to dynamically select a specific language column from a table based on the user's input. Currently, some of our model query subjects use a centralized macro (created earlier in this module) to dynamically select a language column based on the user's locale. You will change this centralized macro to allow the user to specify the data language.

1. In **Framework Manager**, from the **Project** menu, click **Session Parameters**.

2. In the **Value** field for the **UserLanguage** model session parameter, change the existing macro from `#$Language_lookup{$runLocale}#` to **`#prompt('Select language of preference','token','EN')#`**.

This macro passes SQL syntax to append the language code value to the column name the user wishes to select. If no value is provided, the default value is set to EN, which will return data in English.

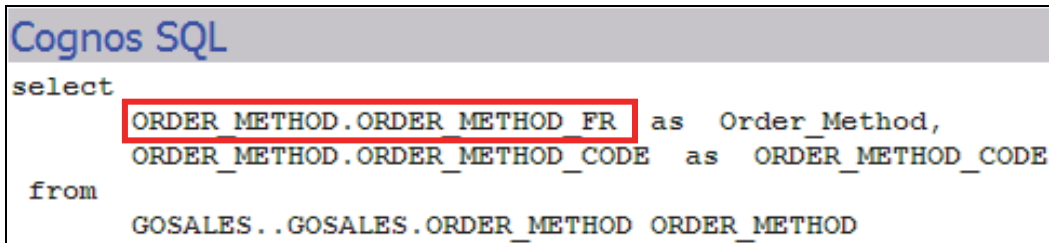
This is a good example of how a centralized macro can save time. This one change will be reflected throughout the model.

3. Click **OK**.
4. In the **Project Viewer**, in the **Consolidation View**, right-click **Order Method**, click **Test**, and then click **Test Sample**.
5. In the **Prompt Values** dialog box, specify a value of **FR**, ensure **Always prompt for values when testing** is selected, and then click **OK**.

Notice that the values in the Order Method column are in French.

6. Click the **Query Information** tab.

The results appear as shown below:



```

Cognos SQL
select
  ORDER_METHOD.ORDER_METHOD FR as Order_Method,
  ORDER_METHOD.ORDER_METHOD_CODE as ORDER_METHOD_CODE
from
  GOSALES..GOSALES.ORDER_METHOD ORDER_METHOD

```

Notice that the Order Method column is suffixed with FR. The FR at the end of this column name is the value you provided and was appended to the SQL at run time.

7. Click the **Test** tab, click **Test Sample** again, clear the **FR** prompt value, and click **OK**.

The results are returned in English because the prompt macro is supplying the default value of EN.

8. Click **Close**.

Task 2. Apply a prompt macro in the data source properties.

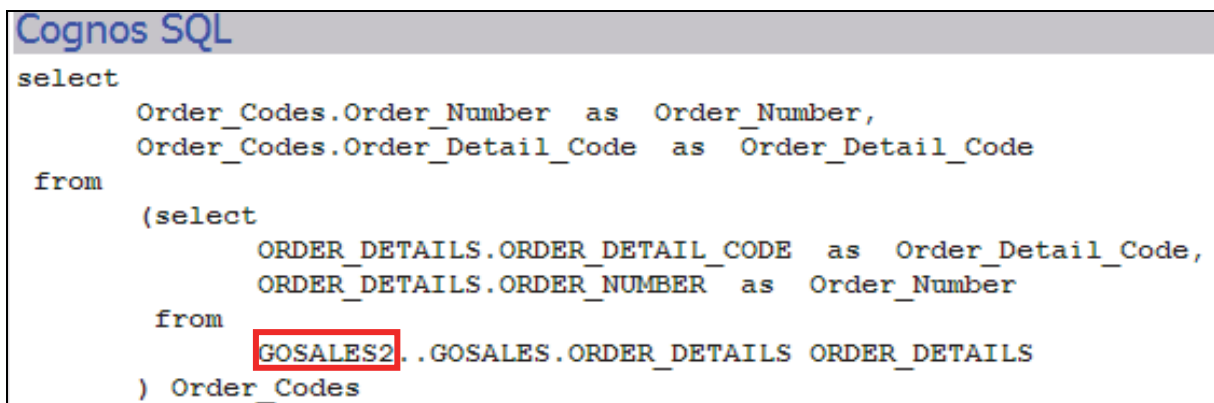
1. Select the **GOSALES** data source.
2. In the **Properties** pane, for the **Content Manager Datasource** property, change the macro to the following:

#prompt('Enter data source number', 'int', 'GOSALES', 'GOSALES')#

This prompt syntax requests a number value from the user and appends it to the preceding text specified, in this case, GOSALES. If no value is specified, then the default value of GOSALES will be used.

3. Click **OK**, and then test **Order Codes**.
4. In the **Prompt Values** dialog box, specify a value of **2**.
5. Click **OK**, and then click the **Query Information** tab.

The results appear as follows:



The screenshot shows a Cognos SQL window with the following SQL query:

```
select
    Order_Codes.Order_Number as Order_Number,
    Order_Codes.Order_Detail_Code as Order_Detail_Code
from
    (select
        ORDER_DETAILS.ORDER_DETAIL_CODE as Order_Detail_Code,
        ORDER_DETAILS.ORDER_NUMBER as Order_Number
    from
        GOSALES2.ORDER_DETAILS ORDER_DETAILS
    ) Order_Codes
```

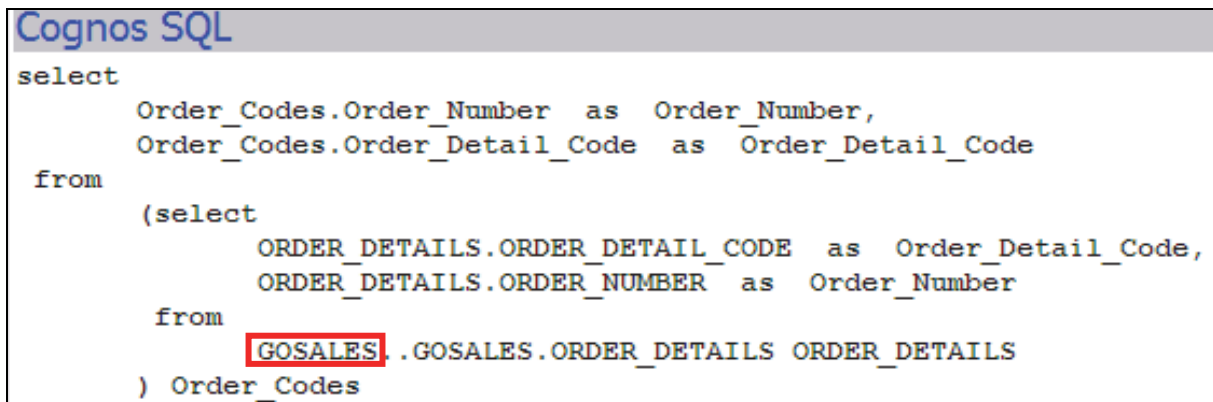
The text **GOSALES2** in the query is highlighted with a red box, indicating the result of the prompt macro.

Notice the data source name is GOSALES2. The value of 2 provided in the prompt was appended to the preceding text GOSALES. This query has gone against the GOSALES2 data source.

6. Test the data again, but this time clear the value of **2** and leave it blank.

7. Click the **Query Information** tab.

The results appear as follows:



```

Cognos SQL
select
    Order_Codes.Order_Number as Order_Number,
    Order_Codes.Order_Detail_Code as Order_Detail_Code
from
    (select
        ORDER_DETAILS.ORDER_DETAIL_CODE as Order_Detail_Code,
        ORDER_DETAILS.ORDER_NUMBER as Order_Number
    from
        GOSALES.ORDER_DETAILS ORDER_DETAILS
    ) Order_Codes
  
```

Notice the data source name is now GOSALES. The default value of GOSALES is used. This query has gone against the GOSALES data source. If you provided a value of anything other than 2 at this point would return an error since there is no data source configured other than GOSALES and GOSALES2.

8. Click **Close**.

Note: Again, you can achieve the same effect in IBM Cognos Connection by giving a user access to more than one connection associated with a Data Source defined. If a user has permissions to access more than one connection for a data source, they will be prompted at run time to select a connection.

Task 3. Apply a filter to the Return Reason Dimension query subject.

1. In **Consolidation View**, double-click the **Return Reason** model query subject to open the **Query Subject Definition** dialog box.
2. Click the **Filters** tab, and then create a filter called **Prompt Many for Return Reason** with the following syntax:

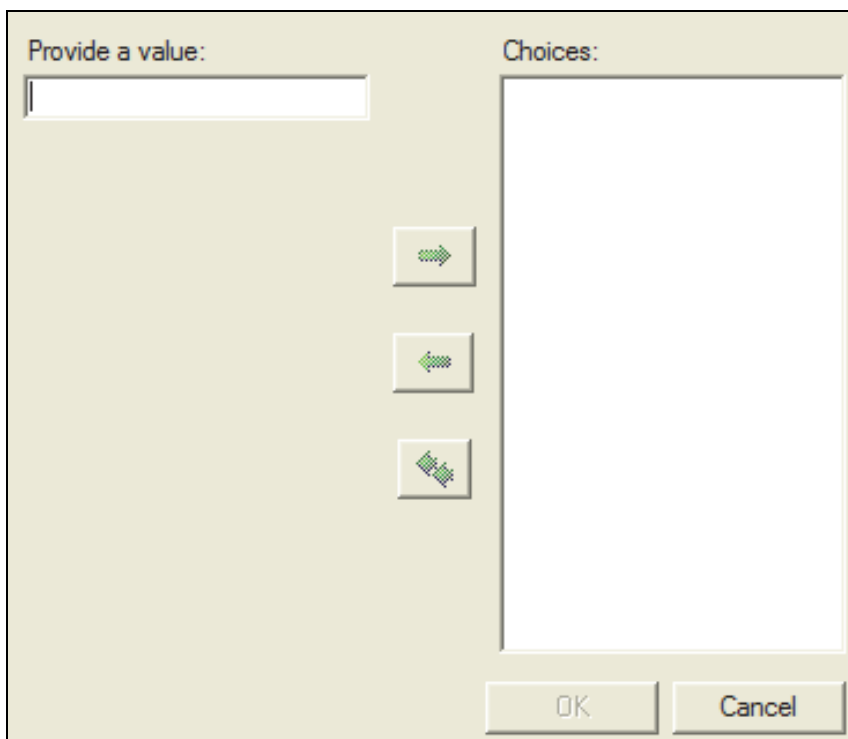
[Consolidation View].[Return Reason].[Return Reason Code] in (#promptmany('Enter Return Reason Codes(s)', 'integer')#)

3. Click **OK**, click the **Test** tab, and then click **Test Sample**.


You are prompted to enter values.

4. Click the **Value** field for **Enter Return Reason Code(s)**.

The results appear as follows:

A screenshot of a multi-value prompt dialog box. The dialog has a light beige background. On the left, under the label "Provide a value:", there is a text input field. In the center, there are three buttons with green arrows: a right-pointing arrow (Add), a left-pointing arrow (Remove), and a double-headed arrow (Clear). On the right, under the label "Choices:", there is a large, empty list box. At the bottom of the dialog are two buttons: "OK" and "Cancel".

A multi-value prompt dialog box appears.

5. In the **Provide a value** box, type **1**, and then click the right arrow  to add the value to the **Choices** list.
6. Repeat to add the values **3** and **5**.
7. Click **OK**, click **OK** again.

The results appear as follows:

Test results	
Return Reason Description	Return Reason Code
Defective product	1
Wrong product ordered	3
Unsatisfactory product	5

Only the values you typed are returned. This is a mandatory prompt since no default values have been provided. If you do not provide a value, you cannot proceed beyond the prompt.

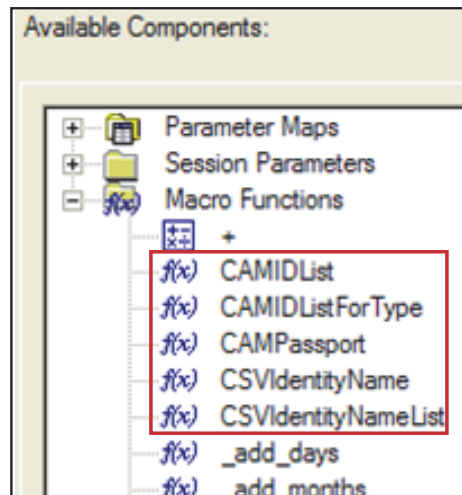
8. Click **OK**, and then save the project.

Results:

You implemented three filters using prompt macros, allowing report authors to easily refine their reports or select their data source.

Security Macro Functions

- Five macro functions associated with security



The macro functions above retrieve security related information from a user's session and can be used in a macro to filter data or display user information in a calculation.

CSVIdentityName

- Retrieves account, group, and role information for the current user
- Use with a parameter map to implement row-level security based on values stored in a data source

[Business view].[Employee].[Security_Access] — in

(#CSVIdentityName(%SecurityLookup)#)

Parameter Map

Key	Value
Americas	NA
Europe	EU
Australia	AU

Database Table

Country	City	Security_Access
Canada	Toronto	NA
Germany	Frankfurt	EU
United States	New York	NA

Cognos.
software

© 2010 IBM Corporation

You can use the CSVIdentityName macro function as a key in a parameter map.

CSVIdentityNameList

- Returns a separator delimited list of the account, group, and role information for the current user.

[Business view].[Retailer site].[City] in
(#CSVIdentityNameList()#)

Database Table

Country	City
Canada	Toronto
Germany	Frankfurt
France	Lyon

Resulting
Filter

[Business view].[Retailer site].[City] in
('Everyone', 'Authors', 'Lyon')

Cognos.
software

© 2010 IBM Corporation

You can use the CSVIdentityList macro function to filter on data that has the same name as the account, group or role information returned in the delimited list.

In the slide example, the city column would be filtered on Lyon.

Demo 5: Leverage a Macro Function Associated with Security

Purpose:

A request has come in to let authors filter their reports based on retailers located in their city. The IT department has set up an LDAP authentication provider with groups based on city names and has added the appropriate users to each group. With this knowledge, you can leverage the group names from the authentication provider in your filter through the CSVIdentityNameList macro function.

There is also a requirement for authors to obtain their personal IBM Cognos security information. To accomplish this, you will create a calculation that will return this information for them.

Components: Framework Manager, Business Insight Advanced

Project: GO Operational

Package: GO Operational (query)

Task 1. Create a filter using the CSVIdentityNameList macro function.

1. In **Framework Manager**, in the **Presentation View** namespace, right-click the **GO Operational Sales (query)** namespace, create a new filter called **Retailers Near Me**.
2. In the **Available Components** pane, expand **Consolidation View>Retailer by Location**, and then add **Retailer City** to the expression definition.
3. In the **Expression definition** pane, after **[Consolidation View].[Retailer by Location].[Retailer City]**, type in **(**.
4. Under **Available Components**, click the **Parameters** tab, and then expand **Macro Functions**.

- Double-click **CSVIdentityNameList** to add it to the expression definition, and then type `)` to finish the expression.

The results appear as follows:

```
[Consolidation View].[Retailer by Location].[RTL_CITY] in
(#CSVIdentityNameList())
```

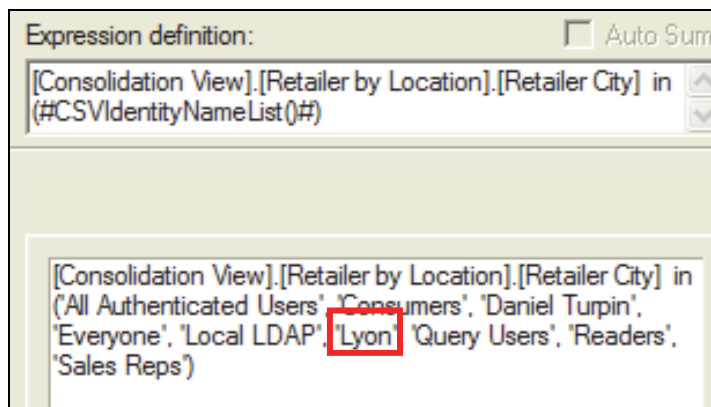
Notice the comma-separated list that appears in the Tips pane.

```
[Consolidation View].[Retailer by Location].[RTL_CITY] in('Admin Person', 'All
Authenticated Users', 'Cognos', 'Consumers', 'Everyone', 'LDAP', 'Readers', 'System
Administrators' )
```

This list is alphabetical and indicates the user's name and all the groups and roles to which they belong. The test user you will use is Daniel Turpin who belongs to a group called Lyon named after the city where he works.

- Click **OK**, and then log off and log back on with as **turpind** (password **Education1!**).
- Double-click the **Retailers Near Me** filter.

The results appear as follows:



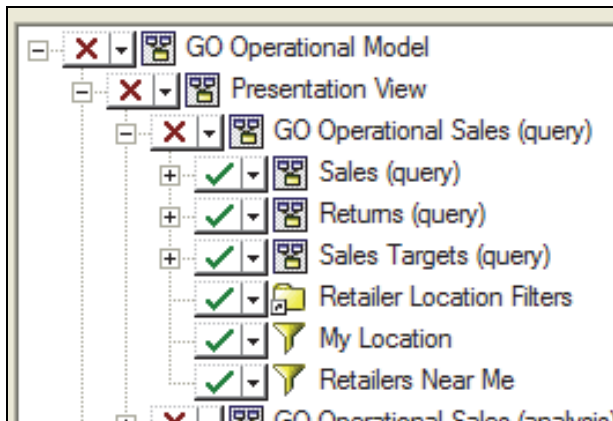
Notice the group called Lyon.

- Click **OK**, log back on as **admin** (password = **Education1!**), and then save the project.

Task 2. Publish the GO Operational (query) package and test it in Business Insight Advanced.

1. Under **Packages**, double-click **GO Operational (query)**, and then select **Retailers Near Me**.

The results appear as follows:



2. Click **OK**, and then publish the **GO Operational (query)** package.
3. In **IBM Cognos Connection**, log on as **turpind** (password = **Education1!**).
4. Launch **Business Insight Advanced** selecting the **GO Operational (query)** package for a **List** report.
5. Expand **Sales (query)**, and then add the following items to the report:

Query Subject	Query Item
Retailer by Location	Retailer City
	Retailer Name
Sales Fact	Revenue

6. Group the report on **Retailer City**.

7. Drag the **Retailers Near Me** filter onto the report.

The results appear as follows:

Retailer City	Retailer Name	Revenue
Lyon	Air marin	\$8,248,477.50
	Amis de montagne	\$3,689,130.62
	Amisport	\$5,395,281.79
	Camping Sauvage	\$4,704,758.74
	Conception française	\$17,175,057.69
	Cordages Discount	\$3,119,130.38
	Galerie Sport	\$8,154,265.75
	Golf Plaza	\$5,745,840.93
	Golf Plus	\$4,270,750.10
	Jeunesse active	\$4,962,985.15
	Monde de sport	\$6,525,600.69
	SportsClub	\$4,151,322.56
	Équipement campant	\$4,450,084.57
Lyon - Summary		\$80,592,686.47

The report is now filtered on Lyon since that is the group to which Daniel Turpin belongs.

8. Return to **IBM Cognos Connection** without saving the query.

Task 3. Create a calculation to retrieve a user's IBM Cognos security information.

1. In **Framework Manager**, right-click the **GO Operational Sales** (query) namespace, point to **Create**, and then click **Calculation**.

2. In the **Name** box, type **My Security Info**.

You will use the sq (single quote) function, the csv (comma separated values) function, and the CAMIDListForType function to return the user's roles. The sq function places the entire results in single quotes making it a string. The csv function takes the results of the CAMIDListForType function and separates the values with a comma. And finally the CAMIDListForType retrieves specific security information for the user based on a parameter. You can request the user's roles, groups, or account.

3. Use the **Parameters** tab to create the following expression:

#sq(csv(CAMIDListForType('role')))#

In this expression, 'role' has been hard coded into the CAMIDListForType function to return the user's roles.

4. Click the **Test Sample**  button.

The results appear as follows:

Test Results	
My Security Info	
'CAMID('::Consumers')','CAMID('::Readers')','CAMID('::System Administrators')'	

You are currently logged on as the admin user. This user belongs to the System Administrators and Readers roles. You can change the 'role' value in the expression to 'group' or 'account' to retrieve different security information.

5. Click **OK**, save the project.

Task 4. Test the security information calculation in Business Insight Advanced.

1. Add the **My Security Info** calculation to the **GO Operational (query)** package definition, and then publish the package.
2. In **IBM Cognos Connection**, launch **Business Insight Advanced**, and then select the **GO Operational (query)** package for a **List** report.

At this point, you are still logged on as turpind in IBM Cognos Connection.

3. In **Business Insight Advanced**, drag the **My Security Info** calculation to the work area.

This user belongs to three roles, the Query Users role, Consumers role and the Readers role.

4. Return to **IBM Cognos Connection** without saving the query.

Results:

Using the CSVIdentityNameList macro function, you created a model filter that lets authors filter their reports based on retailers in the city where they live and work.

You also created a calculation to return users' security information.

Workshop 1: Make the Security Information Calculation Dynamic

Component: **Framework Manager, Business Insight Advanced**

Project: **GO Operational**

Package: **GO Operational (query)**

Currently the My Security Info calculation is hard coded to display a user's roles. Users would like the flexibility to show different information such as their account info or groups they belong to.

To accomplish this, you will need to add a prompt macro to the calculation expression. Make sure the prompt name indicates the appropriate choices for values to be entered into the prompt (role, group, account) and that you specify the correct data type (the values submitted by the user will be used in a SQL query to the Content Store database).

Make the change to the My Security Info calculation, test in Framework Manager, and then test in Business Insight Advanced as turpind (password = Education1!) using the GO Operational (query) package.

Once finished you can save the project and close Framework Manager.

For more detailed information outlined as tasks, see the Task Table on the next page.

For the final results, see the Workshop Results section that follows the Task Table.

Workshop 1: Task Table

Task	Where to Work	Hints
1. Add a prompt macro function to the My Security Info calculation.	Project Viewer pane, Calculation Definition dialog box	<ul style="list-style-type: none"> Edit the My Security Info calculation definition to appear as shown below: <code>#sq(csv(CAMIDListForType(prompt ('Specify info type (role, group, account)', 'token'))))#</code> Test three times, supplying a different prompt value each time (role, group, account) to see the effects of the prompt values. Save the project, and then publish the GO Operational (query) package.
2. Test the My Security Info Calculation in Business Insight Advanced.	Business Insight Advanced	<ul style="list-style-type: none"> In IBM Cognos Connection logged in as turpind (password=Education1!), launch Business Insight Advanced selecting the GO Operational (query) package for a List report. Drag the My Security Info calculation onto the report, and then type the word group in the prompt value box. In Framework Manager, close the project, saving if prompted.

Workshop 1: Results

You will see the following prompt when testing the calculation in Framework Manager:

Enter prompt values:		
Name	Data Type	Value
* Specify info type (role, group, account)	String (Abc)	group

In query studio, the prompt should appear as follows:

Prompt

Provide values for the report you are about to run.

- * Indicates a required field.
- ➔ Points to missing information.

Specify info type (role, group, account)

Provide a value:

* ➔

If you specify group as your prompt value, the results will appear similar as follows:

```
'CAMID("Sales Reps") ', 'CAMID("::All Authenticated
Users") ', 'CAMID("::Everyone") ', 'CAMID("Local LDAP
ID:g:23edba01-8f2f11dd-80c78a25-58ae3caf") '
```

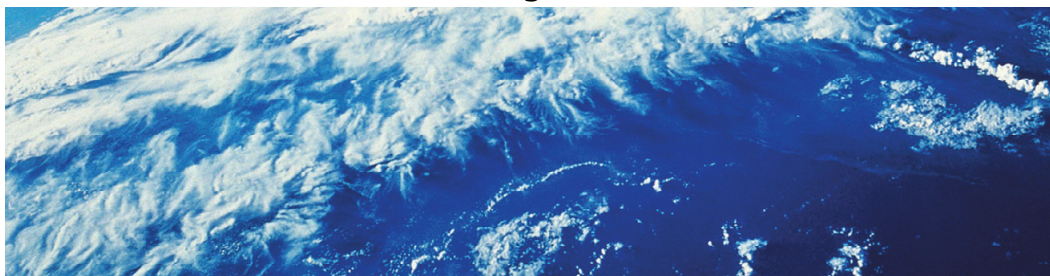
Summary

- You should now be able to:
 - identify session and model parameters
 - leverage session, model, and custom parameters
 - create prompt macros
 - leverage macro functions associated with security



Model Maintenance and Extensibility

IBM Cognos BI



Business Analytics

© 2010 IBM Corporation

Objectives

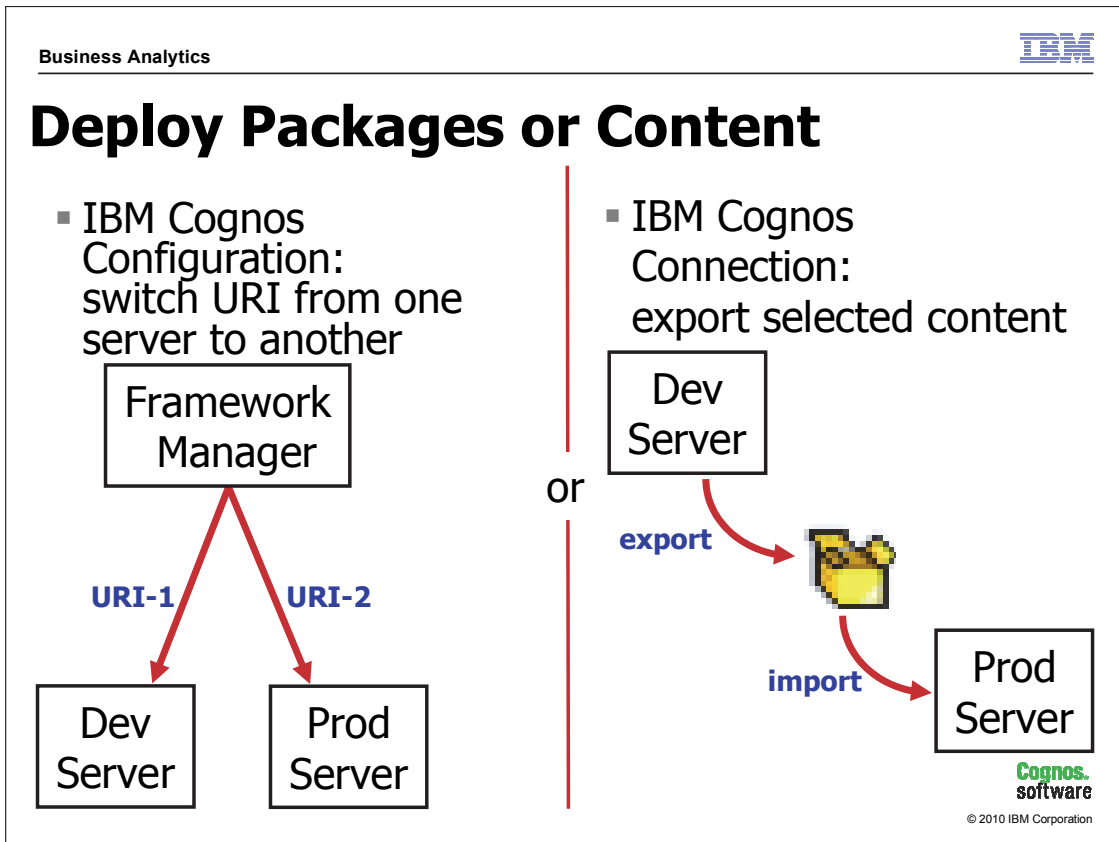
- At the end of this module, you should be able to:
 - perform basic maintenance and management on a model
 - remap metadata to another source
 - import and link a second data source
 - run scripts to automate or update a model
 - view lineage
 - create a model report

Copy, Move, Rename and Delete Projects

- These actions:
 - are available from File > Manage Projects
 - allow you to organize projects in a meaningful way
 - work on the file folder and all contents (except Rename)
 - require closing the project first (except Copy)

Rename Project renames the .cpf file but no other files in the folder. If the folder has the same name, it is also renamed. Of course these same actions can be performed manually on the system files in Windows Explorer.

If the project has segments, Copy, Move, and Delete work on all segments and the main project. However deleting a project segment does not delete the main project or other segments. Segments are discussed in more detail in another module.



When you publish a package, Framework Manager uses the Dispatcher URI (Uniform Resource Indicator) setting in IBM Cognos Configuration (Environment – Group Properties) to determine the target server. A simple way to deploy a package to a different server is to change this setting in IBM Cognos Configuration. This only allows you to deploy a package to a different environment, not content (reports, analysis, and so on) based on that package. For example, reports created in the development environment will not be deployed to the production environment using this method.

To deploy packages and their related content, an administrator can create an Export Deployment Specification to export selected packages, reports, security objects, and other objects from the content store. These can then be imported to other IBM Cognos servers (such as multiple production sites). You can also create an export deployment of the entire Content Store.

Analyze Publish Impact

- Framework Manager feature that analyzes the effects of model changes on a published package
- Identifies:
 - changed model objects
 - other model objects affected by the changes
 - reports that use the changed objects

For each object identified as a changed model item, an icon identifies whether the change may or will break any related reports. You can then identify specific reports affected by the changes, and even directly run those reports from the Analyze Change Impact dialog box.

With this information, you can notify report authors that a change was made to the model that affects their reports. They can then fix their reports so that consumers are not affected.

Something to be aware of is that changed cardinality is not detected and identified by this tool since changed cardinality does not break reports. It may, however, change the amount of data returned (outer join vs. inner join), and therefore should be discussed with authors.

Demo 1: Perform Impact Analysis on a Modified Package

Purpose:

A number of reports have been created using the GO Operational model. You need to make some changes to the model in Framework Manager and re-publish the package. Before publishing, you want to analyze what effect this change will have on existing reports. To accomplish this you will perform an impact analysis on the GO Operational package.

Component: **Framework Manager**

Project: **GO Operational**

Task 1. Modify a model object.

1. In **Framework Manager**, close any projects that may be open, and then open the **GO Operational** project located at **C:\Edcognos\B5152\CBI-FM-Start File\Module 20\GO Operational**.
2. If prompted, log in as User ID **admin**, and Password **Education1!**.
3. Publish the **GO Operational (query)** package.
4. In the **Project Viewer** pane, under **Consolidation View**, rename **Time** to **Time (Sale)**.

Task 2. Analyze the impact of changing the model.

1. Under **Packages**, click the **GO Operational (query)** package.
2. Right-click the **GO Operational (query)** package, and then click **Analyze Publish Impact**.

The Analyze Publish Impact dialog box appears.

3. In the **Changed Model Items** section, click **Time (Sale)**.

The results appear as follows:

Save Print

Publish Impact for Package: GO Operational (query)

The list below shows the package changes relevant to the reporting environment. A change in a package may have an effect on reports that reference it.

Changed Model Items: [Find Report Dependencies](#)

<input type="checkbox"/>			Object Name	Change	Actions
<input type="checkbox"/>			Time	Modified	
<input type="checkbox"/>			Time	Modified	
<input type="checkbox"/>			Time	Modified	
<input type="checkbox"/>			Time (Sale)	Modified	

Change details for:
List of the relevant model item properties that have changed.

Property	Old Value	New Value
name[en]	Time	Time (Sale)

Notice the red X icon, which indicates that the change you made will break related reports. The "Change details for" section identifies the nature of the change, in this case the renaming of Time to Time (Sale).

Under Actions, you can show the object dependencies , find the object in the Project Viewer , or open the objects definition .

4. To the right of **Time (Sale)**, under **Actions**, click **Show Dependencies** , if necessary, move the **Analyze Publish Impact** window to view the **Tools** pane.

The Dependencies tab identifies the modified query subject and its contents. The Dependent Objects list identifies all objects affected by it, including shortcuts, model query subjects and query items.

5. In the **Analyze Publish Impact** window, to the left of **Time (Sale)**, select the check box and then, above **Actions**, click **Find Report Dependencies**.

You can choose the scope of your dependencies search.

6. Click **Search**.

A list of reports affected by your change is returned. Because you haven't saved many class reports, making this change has little effect. You can identify the report the object appears in and how it is being used. Of course, in a production system this could identify hundreds of reports, indicating that it might not be practical to make this change now. You could enable model versioning to allow authors time to repair their reports. Model Versioning is discussed in another module.

You can click on any of the report names in the list to view and edit the report in IBM Cognos.

7. Click **Close**, and then click **Close** again.
8. Click **Undo** to restore the original name to **Time**, and then save the project.

Results:

You made a change to the GO Operational model in Framework Manager. Before you re-published, you analyzed the package to determine what effect this change to the model would have on existing reports.

Remap An Object to a New Source

- Useful when underlying objects have changed
- Remaps objects to new items
- Typically used to remap middle layer(s) to new or changed data source items
- Remaps automatically (based on name matching or by object reference) or manually (drag and drop)

The remapping feature can be very useful in remapping your consolidation view (middle layer) to new or changed data source objects. For example, you may switch database vendors or database structures (move from an operational structure to a star schema structure).

Remapping is also useful if you decide to change how you modeled an underlying object or objects. For example, if your Product dimension in the Consolidation View references two underlying query subjects (Product and Product Type) and you decide to merge those underlying query subjects, you can use the remap tool to point the Product dimension query subject to the newly merged query subject in the lower layer.

Demo 2: Remap the Physical Layer

Purpose:

You have been given a new and improved data source, **great_outdoors_warehouse**, which will replace the **gosales** data source under **Foundation Objects View**. Your ultimate goal is to remap all **Consolidation View** model query subjects to data source query subjects in the new data source. You will start this by remapping the **Consolidation View's Time** query subject to point to the new **GO_TIME_DIM**.


Component: **Framework Manager**

Project: **GO Operational Maintenance**

Task 1. Import the new metadata.

You will create another version of your model to perform the remapping technique. This will allow you to maintain a backup copy of your original project.

1. From the **File** menu, click **Save As** and save your current project as **GO Operational Maintenance** in **C:\Edcognos\B5152\Course_Project\GO Operational Maintenance**.
2. In the **Project Viewer** pane, in the **Foundation Objects View** namespace, create a new namespace called **GO Data Warehouse**.
3. Right-click **GO Data Warehouse**, and then click **Run Metadata Wizard**.
4. With **Data Sources** selected, click **Next**.
5. Select **great_outdoors_warehouse**, and then click **Next**.
6. Expand **GOSALESDW** and **Tables**, select **GO_TIME_DIM**, and then click **Next**.

7. Click **Import**, and then click **Finish**.
8. Expand **GO Data Warehouse>GO_TIME_DIM**.
9. For all the query items displayed as **Facts** , change the **Usage** property to **Attribute**.

Task 2. Remap Time to GO_TIME_DIM.

Before you remap objects, it is a good idea to find all object dependencies for the object that will be replaced with a new one. In this case it is the original **TIME_DIMENSION** query subject from the **gosales** namespace. To do this you will use the Dependencies feature to identify which objects will require remapping.

1. In the **gosales** namespace, right-click **TIME_DIMENSION**, and then click **Show Object Dependencies**.

In the Tools pane, on the Dependencies tab, **TIME_DIMENSION** and its children appear in the top pane, and dependant objects appear in the bottom pane. Selecting items in the top pane filters the items in the bottom pane. The items in the bottom pane include dependant packages, security, relationships, query items, and determinants. If you scroll down in the Dependant objects list, you can select individual query items. Each one you select receives focus in the Project Viewer tree. This way you can identify which objects will require remapping.

2. In the **Tools** pane, in the **Dependant objects** pane, select the first **Day Key**.
Consolidation View>Time>Day Key is selected in the Project Viewer tree. You now know this object requires remapping to the new **GO_TIME_DIM** query subject. Other objects also require remapping, but for the purposes of this demo, you will only remap one object.
3. In **Consolidation View**, right-click **Time**, and then click **Remap to New Source**.

The right pane identifies the original source for all query items.

4. Under **Available Model Objects**, expand **Foundation Objects View** > **GO Data Warehouse** > **GO_TIME_DIM**.

Comparing left and right panes, it appears that the new query subject uses the same query item names as the original query items. You can drag each query item from the left to the right pane individually, or you can use the matching criteria feature. Note that 'Use matching criteria options' is selected.

5. Click **Options**.

You see that the default for matching criteria is to remap to by name.

6. Click **Cancel**, and then drag **GO_TIME_DIM** to anywhere in the right pane.

Name	Remap To	Original Source
Year	[GO Data Warehouse].[GO_TIME_DIM].[CURRENT ...	[gosales].[TIME_DIM]
Quarter Key	[GO Data Warehouse].[GO_TIME_DIM].[QUARTER ...	[gosales].[TIME_DIM]
Quarter	[GO Data Warehouse].[GO_TIME_DIM].[CURRENT ...	[gosales].[TIME_DIM]
Month Key	[GO Data Warehouse].[GO_TIME_DIM].[MONTH ...	[gosales].[TIME_DIM]
Month (numeric)	[GO Data Warehouse].[GO_TIME_DIM].[CURRENT ...	[gosales].[TIME_DIM]
Month	[GO Data Warehouse].[GO_TIME_DIM].[CURRENT ...	[gosales].[Month]
Day Key	[GO Data Warehouse].[GO_TIME_DIM].[DAY_KEY ...	[gosales].[TIME_DIM]
Date	[GO Data Warehouse].[GO_TIME_DIM].[DAY_DAT ...	[gosales].[TIME_DIM]

A match was found for all query items except one calculation. This must be handled manually. If a match wasn't found because of different names, deselect 'Use matching criteria options', and drag the individual query item from the left pane to the appropriate query item in the right pane.

7. Click **OK**.

This produces a warning that there will be a cross join error, as there is no link between GO_DATA_Warehouse.GO_TIME_DIM and gosales.TIME_DIMENSION. You can ignore this, as you repair the two calculations causing this error.

8. Click **OK**.

Notice that the Time icon in Consolidation View has changed. The jagged break indicates an invalid object.

Task 3. Manually repair calculation.

1. Expand **Foundation Objects View>gosales>Reusable Objects>Model Calculations**, and then double-click **Month**.
2. In the Expression definition, change **[gosales].[TIME_DIMENSION]** to **[GO Data Warehouse].[GO_TIME_DIM]**, click **Test Sample**, and then click **OK**.
3. In **Consolidation View**, click **Time**, and then from the **Tools** menu, click **Update Object**.

This causes Framework Manager to reevaluate the Time object. All query items now point to just one data source query subject, which eliminates the cross join error. The query subject icon is restored to its normal appearance.

In a non-classroom environment, you would continue to do similar remapping for all the other Consolidation View objects. This work would involve handling any reporting traps in the new GO Data Warehouse namespace before remapping. Once you have completely replaced references to gosales with references to GO Data Warehouse, you would delete the gosales namespace. You would also move the Reusable Objects folder into the GO Data Warehouse namespace. In the interest of time, you will skip these steps.

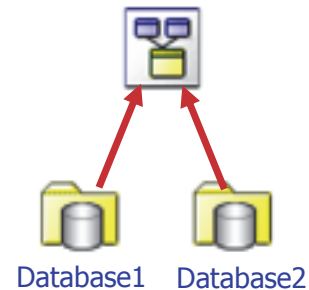
4. Save the project.

Results:

You have remapped your Time model query subject to a new data source query subject, and are ready to continue the remapping from gosales to GO Data Warehouse.

Link Multiple Data Sources

- You can import metadata from multiple data sources.
- To link the data sources, create relationships between them on common query items.



When linking heterogeneous data sources, you must create relationships on common data to obtain expected results. For example, if you imported an Inventory fact table from another data source that contained a Product Key (matching in values and data type to the Product Key in your existing Product dimension) you can use that key to create a relationship to the Product dimension. By doing so, you can create queries that retrieve data from separate data sources with related results.

Non-Database Data Sources

In addition to using databases as data sources for your models, several file types can be set up as data sources as well, including:

- Microsoft Excel spreadsheets
- Microsoft Word tables
- XML files
- Comma separated value (CSV) files

Often, files will need an ODBC connection through Microsoft Windows or IBM Cognos Virtual View Manager to enable the use of individual files as data sources.

Demo 3: Create a Microsoft Excel Data Source

Purpose:

A manager has an Excel file he uses to identify particular employees he wants to monitor for special projects. He has asked you to set up his pick list as a data source for such purposes.

Component: **Framework Manager, Business Insight Advanced**

Project: **GO Operational**

Task 1. Set Picklist.xls as an ODBC Data Source.

1. Open **C:\Edcognos\B5152\CBIFM-Start Files\Module 20\PickList.xls** in Microsoft Excel.
2. Select all the populated cells, and then from the **Insert** menu, click **Name>Define**.

The Define Name dialog opens.

3. In the **Names in Workbook** text box, type **PickList**, and then click **OK**.
 4. Save the spreadsheet to the desktop and exit Excel.
- Since this is a training environment, we are using the Desktop. In your organization's environment, use the Cognos server.
5. Go to **Start>Control Panel>Administrative Tools>Data Sources (ODBC)**.
 6. Click the **System DSN** tab, and then click **Add**.

The Create New Data Source dialog opens.

7. Select **Microsoft Excel Driver (.xls)**, and then click **Finish**.
8. In the **Data Source Name** text box, type **PickList**, and click **Select Workbook**.
9. Navigate to the Desktop, select **PickList.xls**, click **Open**, and then click **OK** three times.

Task 2. Import PickList into the Project.

1. Return to Framework Manager.
2. Right-click the **Foundation Objects View** namespace, point to **Create**, and then click **Namespace**.
3. Rename **Namespace** to **PickList**.
4. Right-click the **PickList** namespace, and then click **Run Metadata Wizard**.
5. Ensure that **Data Sources** is selected, and click **Next**.
6. Click **New**.
7. Click **Next**.
8. In the **Name** box, type **PickList**, and click **Next**.
9. In the **Type** list, select **ODBC**, leave the default isolation level, and then click **Next**.
10. In the **ODBC data source** text box, type **PickList**.
11. Scroll to the bottom of the page and select the **No authentication** radio button.
12. Click **Test the connection**, and then click **Test**.
The View the results - Test the connection page appears indicating that the test succeeded.
13. Click **Close**, and then click **Close** again.
14. Click **Finish**, and then click **Close**.
15. Ensure the newly created **PickList** data source is selected, and then click **Next**.
16. In the list of objects, expand **PickList>Tables>PickList**.

17. Select the following initial tables:

Staff Name

Sales staff code

Branch Code

18. Click **Next**, leave the defaults for the **Generate Relationship** criteria, and then click **Import**.

The import process begins, and then a message appears summarizing the seven query subjects and six relationships that were imported.

19. Click **Finish**.
20. In the **PickList** query subject, change the **Usage** property of **Sales staff code** to **Identifier**.

Task 3. Build relationships.

1. In the **Project Viewer** pane, select the following items:
 - **PickList/PickList>Sales Staff Code**
 - **gosales/Sales fact>SALES_STAFF_CODE**
2. Right-click one of the selected items, and then click **Create Relationship**.
3. Click **OK**.
4. Create the following relationships, clicking **No** if asked to replicate the existing underlying relationships:
 - **PickList (Sales staff code, 1..1) to SALES_TARGET (SALES_STAFF_CODE, 1..n)**
 - **PickList (Sales staff code, 1..1) to Returns Fact (SALES_STAFF_CODE, 1..n)**

You now have added the metadata for your Microsoft Excel PickList spreadsheet to your model and linked it to the rest of the metadata in your model.

5. Publish the **GO Operational** package.

Task 4. Create an ad hoc query.

1. Open **Business Insight Advanced** and select the **GO Operational** package for a **List** report.
2. In the **PickList** namespace, expand the **PickList** query subject, and then double-click the **Staff name** query item.
3. Repeat step 2 to add the following items from the **gosales** namespace:

Query Subject	Query Item
Sales Fact	Revenue
SALES_TARGET	SALES_TARGET
Returns Fact	Return Quantity

Only the Sales Reps listed in the Pick List display in the report.

4. Close **Business Insight Advanced** without saving the report.
5. Save the project.

Results:

You have connected to a new data source, imported the metadata, and then built a relationship between the query subjects of the two data sources in your model.

Play Back Parts of an Action Log

- All modeling actions are written to an action log file.
- Save specific recorded actions to play back later.
- Useful for:
 - repeating identical sequences within a model
 - creating a starting point for multiple similar models
 - moving a model change from test environment to production
 - auditing or troubleshooting
 - bulk upgrades in batch mode

Each sequence of actions that you perform in Framework Manager is considered a transaction. Each transaction is recorded in the action log file. You can view the history of transactions, and play back individual transactions or a combination of transactions in a log file. You can also save transactions to a separate log file (script).

The action log file is an XML file that is stored in the project folder.

When troubleshooting a model with IBM Cognos Customer Support, the technical analyst will likely request the log files to be sent in.

Synchronize Projects

- Use log files to synchronize a project with changes in the metadata source.
- All actions performed in the model are replayed. The result is:
 - a new project is created
 - metadata is re-imported to capture any changes
 - the entire modeling process is repeated using updated metadata

The Synchronize feature is found in the Project menu of Framework Manager.

You can choose to accept the new changes and create a new project, or return to the original project. If you accept the new changes, the original project is deleted.

The Synchronize dialog box includes a check box to back up the original project before synchronizing. We recommend that you select this check box.

Check a Project

- High-level: Verify selected objects
 - identifies invalid relationships, references, and object definitions
 - offers automated or manual repairs
- Low-level: Run model advisor
 - identifies design issues around relationships, determinants, and other model factors
 - offers direct links to online help for resolving issues

These two tools have already been used and examined earlier in the course and are options available by right-clicking any objects in your model. They can be invoked at any time in the modeling process.

Demo 4: Run a Script to Replay Actions

Purpose:

You are making some changes to your project in a test environment. You will be adding a Business view folder and adding a query subject to it. You have a corresponding model in a production environment, but it does not contain the changes you are making. To save development time, you will create a script based on the actions performed on the project in the test environment, and then run that script on the project in the production environment to apply those changes.

Component: Framework Manager

Project: GO Operational Maintenance

Task 1. Model a business view for report authors in the test environment.

1. In the **Project Viewer** pane, in the **GO Operational Model** namespace, create an empty folder called **Business View**.
2. Right-click the **Business View** folder, point to **Create**, and then click **Query Subject**.
3. In the **Name** box, type **Retailers**, leave the default selection to define the data to be used to create the query subject, and then click **OK**.
4. In the **Available Model Objects** pane, expand **Foundation Objects View>gosales**.

5. Add the following items to the **Query Items and Calculations** pane:

Query Subject	Query Item
RETAILER_TYPE	TYPE_NAME_EN
Retailer & Retailer Site	RTL_ADDRESS1 RTL_ADDRESS2 RTL_CITY RTL_COUNTRY_CODE

6. Click **OK**.

The Retailers query subject now appears within the Business view folder.

7. Save and close the project.

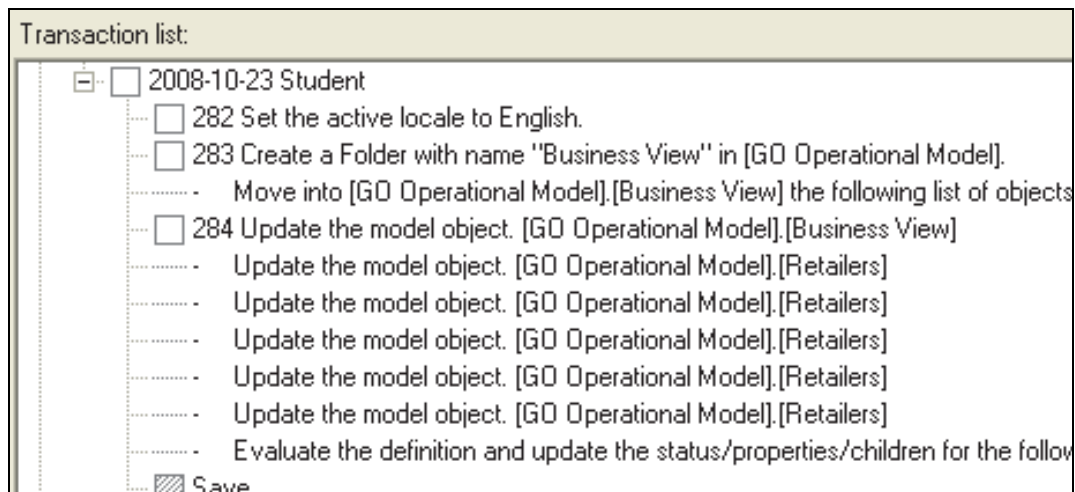
This updates the log file that has been generated for this session. If you wish, you can examine the timestamp for this projects log.xml file in Windows Explorer.

Task 2. Create a script for the creation and population of the Business View folder.

1. Under **Recent Projects**, open the **GO Operational Maintenance** project.
2. From the **Project** menu, click **View Transaction History**.
3. Under **Transaction list**, expand **C:\Edcognos\B5152\Course_Project\GO Operational Maintenance\log.xml** entry (first entry in the list).

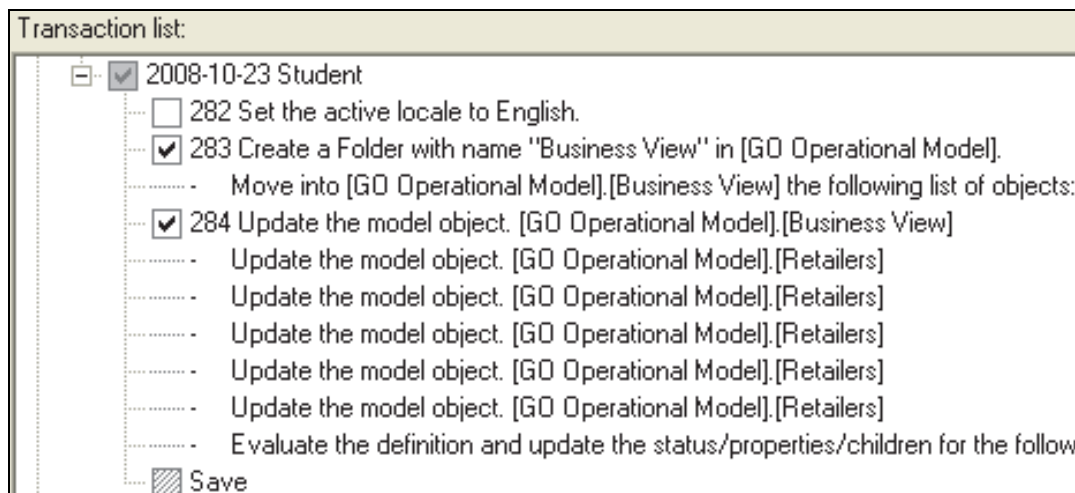
4. Scroll down to the bottom of the entry and then expand the last child entry.

The results appear as follows:



5. Select '**Create a folder with name Business View**' and '**Update the model object**' entries.

The results appear as follows (preceding numbers may be different):



6. Click **Save as Script**.
7. In the **File name** box, type **Create Business View**, and then click **Save**.
8. Click **Close**.
9. In **Project Viewer**, expand the **GO Operational Model** namespace, click the **Business View** folder, and then press **Delete**.

Task 3. Run the script in the production environment to recreate the Business view folder.

You will now assume that you are in the production environment. You will recreate the Business View folder using the script you created in the test environment so that it appears in the production environment.

1. From the **Project** menu, click **Run Script**, and then double-click **Create Business View.xml**.
2. Ensure that all check boxes are selected, and then click **Run**.

A status message quickly appears and then a transaction message appears in the Transaction details pane.

3. Click **Accept**, and then expand **Business View** and **Retailers**.

You successfully recreated the objects using a script.

4. Save and close the project.

Results:

You made changes to your project in a test environment. You created a script to record those actions, and you ran that script on your project in the production environment. The result is that all actions taken on your project in the test environment were applied, and are now reflected in the production environment.

Lineage

- Lineage information traces an item's metadata back through the package and the package's data sources.
- Two types:
 - IBM Cognos-only
 - Third-party provider with a URL interface
- Accessed:
 - in Cognos Viewer (HTML reports only)
 - in the data tree in the studios
 - through the IBM Cognos BI SDK

Modelers and authors alike can use lineage in the IBM Cognos studios and Cognos Viewer to trace metadata back to the data source. This can be a useful tool in helping authors and modelers work together to refine the model to meet requirements.

Lineage can be disabled or enabled on a package-by-package basis through object capabilities.

Business Analytics IBM

Lineage Views

Business View Business View

Order Invoices - Donald Chow, Sales Person

Description: Generates invoices of sales by Donald Chow.

Owner: Admin Person

Contact:

Package:

Name	GO Sales (query)
Description	Package based on relational view.

Report Item 1

Name: Price

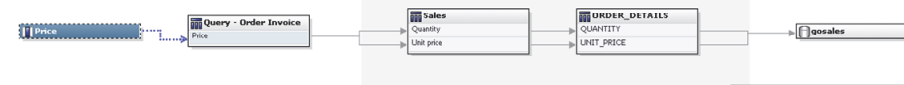
Path: Order Invoices - Donald Chow, Sales Person > Query - Order Invoice > Price

Expression: [Sales (query)][Sales].[Quantity] * [Sales (query)][Sales].[Unit price]

Referenced Package:

Name	Quantity
Path	Order Invoices - Donald Chow, Sales Person > GO Sales (query) > Sales (query) > Sales > Quantity

Technical View Technical View

Report: 

Properties for Price

Property	Value
ID	[Query - Order Invoice].[Price]
Name	Price
Type	Data Item
Expression	[Sales (query)][Sales].[Quantity] * [Sales (query)][Sales].[Unit price]
Datatype	Decimal
Precision	30
Scale	2
Regular Aggregate	Sum

Cognos. software

© 2010 IBM Corporation

Users with access to any of the studios will have access to two tabs in the lineage results, a Business View tab and a Technical View tab. Consumers will only see the Business View results.

Business View - contains a textual, abbreviated form of lineage that is intended to provide some basic information to an end user such as descriptions, expressions, data source and contact information.

Technical View - provides a graphical representation as well as detailed information about the data flow from the metadata tree or report back to the data source.

Integration with Metadata Workbench

- For IBM Cognos v10.1, we have introduced tighter integration with InfoSphere Metadata Workbench.
- You can now leverage IBM Cognos Lineage Viewer for report and package information and can invoke InfoSphere Metadata Workbench to drill down into data source level information
- This new entry point for Metadata Workbench gives you the ability to leverage the best of both products with the following advantages
 - Ease of deployment and maintenance for IBM Cognos Lineage
 - Exploration and analysis power of Metadata Workbench
 - Flexibility to manage access to different levels of information

Demo 5: Explore Lineage

Purpose:

As a consumer, author or modeler, you may like to gain more information about the data reports. You will use the Lineage feature in Cognos Viewer, as well as in the metadata tree of the studios, to retrieve the additional information.

Component: IBM Cognos Connection, Cognos Viewer, Query Studio

Package: GO Sales (query)

Task 1. View lineage from Cognos Viewer.

1. Launch **IBM Cognos Connection**, log on as **admin** (password=**Education1!**), and then click **IBM Cognos content**.
2. Navigate to **Public Folders>Samples>Models>GO Sales (query)>Report Studio Report Samples**.
3. Click **Order Invoices - Donald Chow, Sales Person**.

The report runs and displays in Cognos Viewer. You would like to see how the Price values were derived.

4. Right-click **Price** in the report, and then click **Lineage**.

The results appear as follows:

Business View		Technical View	
Order Invoices - Donald Chow, Sales Person			
Description	Generates invoices of sales by Donald Chow.		
Owner	Admin Person		
Contact			
Package	Name	GO Sales (query)	
	Description	Package based on relational view.	
Report Item 1			
Name	Price		
Path	Order Invoices - Donald Chow, Sales Person > Price		
Expression	[Query - Order Invoice].[Price]		
Referenced Package Items			
Data Sources	Name	gosales	
	Description		

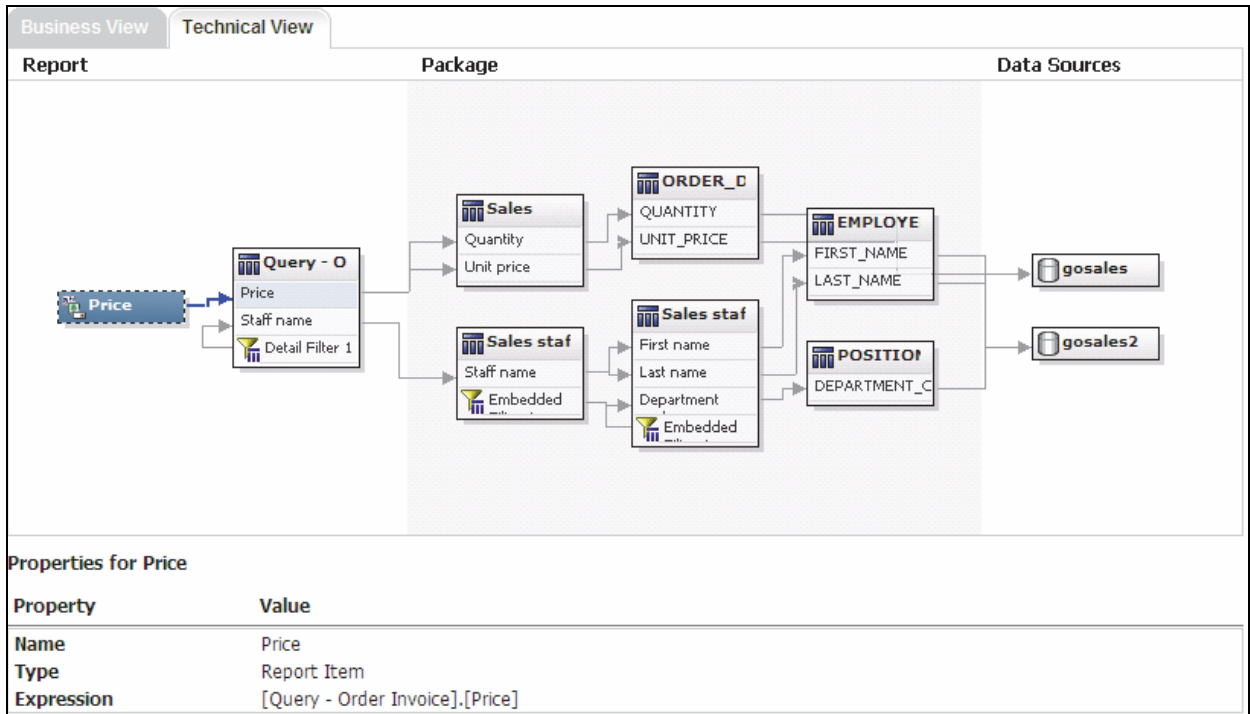
The Business View tab is displayed. If you were logged in as a consumer, you would only see this view and not have access to the technical view.

The Business View displays basic information about the owner of the report, and the name of the package it is based on. For the Price data item, it provides the following information: what its path in the report is, how it was calculated, what the names and paths of the items used in the calculation are, and the name of the data source the items came from. This information typically gives consumers enough information to understand the data they are viewing, or at the very least who to talk to about the report.

For more technical users such as authors, you can view the information on the Technical View tab.

5. Click the **Technical View** tab.

The results appear as follows:

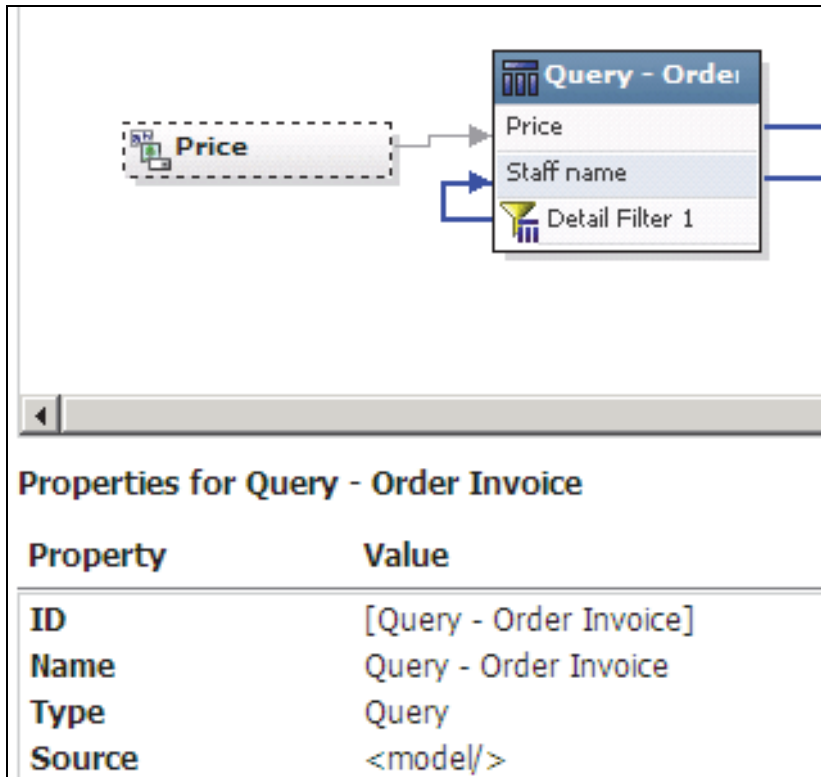


The flow of the metadata from the report back to the data source is displayed. The flow begins in the Report section, showing the item in question and the query it belongs to, followed by the Package section, which shows all relevant model objects, and finally the data source section which indicates the data source name (or names if more than one was used to create the data item in question).

Clicking on any of the items in the diagram will change the information in the Property pane specific to the item that has focus. Currently the Price data item from the report has focus. The information provided includes the nature of the calculation, the datatype, and the aggregate function used, in this case Sum.

6. Click on the **Query - Order Invoice** header.

The Property pane results appear as follows:



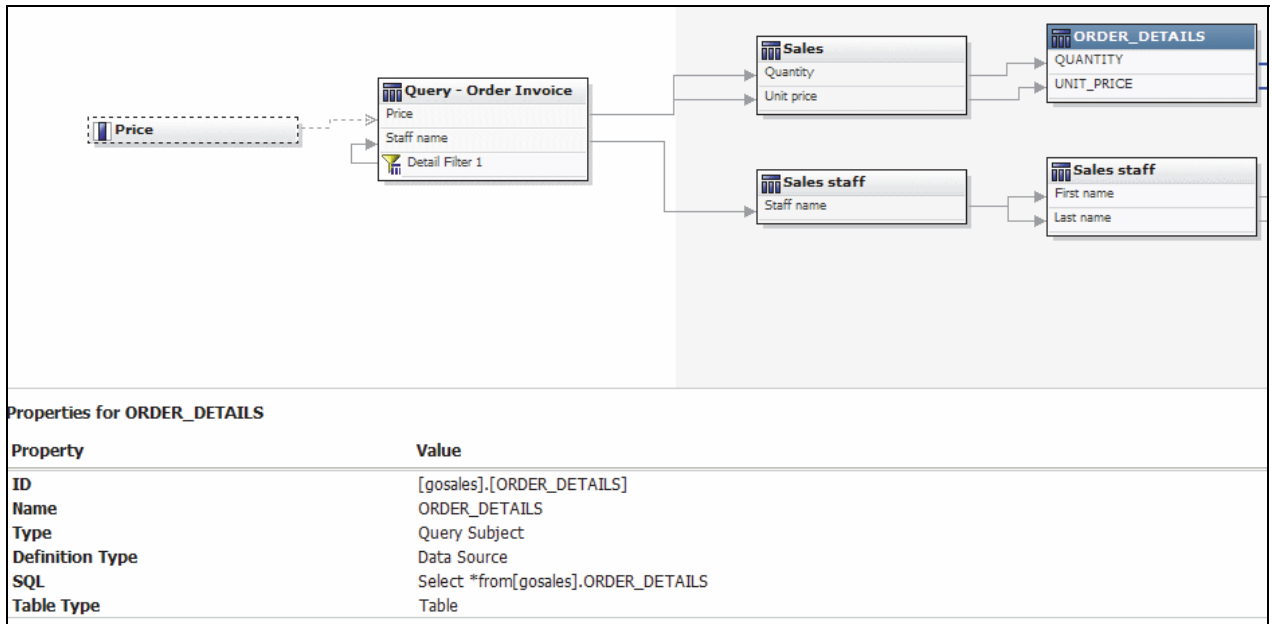
The screenshot shows the IBM Cognos Framework Manager interface. At the top, a header box labeled 'Query - Order Invoice' contains three items: 'Price', 'Staff name', and 'Detail Filter 1'. A dashed box highlights the 'Price' data item, and a blue arrow points from it to the 'Price' field in the header. Below the header, a scroll bar is visible. Underneath the scroll bar, the 'Properties for Query - Order Invoice' pane is displayed, showing a table with the following properties and values:

Property	Value
ID	[Query - Order Invoice]
Name	Query - Order Invoice
Type	Query
Source	<model/>

The property pane now displays information specific to the report query containing the data item Price.

7. In the **Package** section, click on the **ORDER_DETAILS** header.

The results appear as follows:



The Property pane displays information such as the type of query subject (Data Source query subject), and the SQL used to generate the query subject. You can continue to click on any of the items to explore the associated information.

8. Close the **Lineage** window, and then click **Return**.

Task 2. View lineage from the metadata tree in Query Studio.

1. Launch **Query Studio** (it will open using the GO Sales (query) package since you are currently in this package).
2. In the **Insert Data** pane, expand **Product forecast (query)**.

3. Right-click **Product forecast**, and then click **Lineage**.

The results appear as follows:

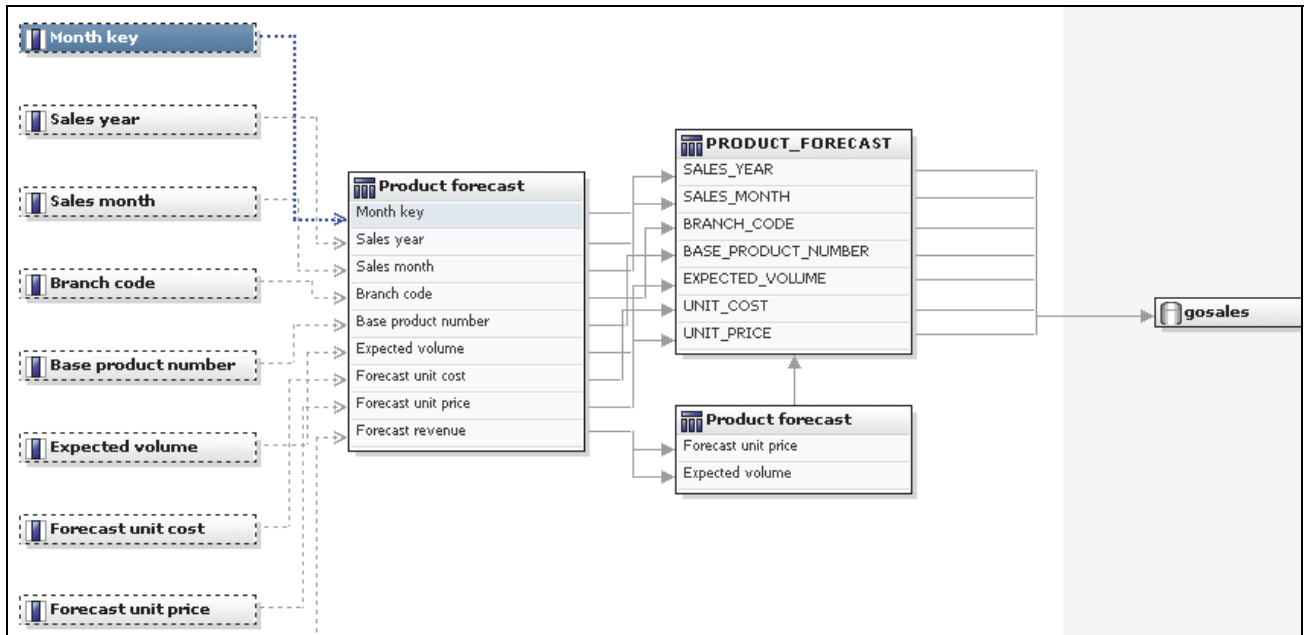
The screenshot shows the 'Business View' tab for a query named 'GO Sales (query)'. It displays the following information:

- Description:** Package based on relational view.
- Owner:** Anonymous
- Contact:**
- Package Item 1:**
 - Name:** Product forecast
 - Path:** GO Sales (query) > Product forecast (query) > Product forecast
 - Description:**
 - Data Sources:**
 - Name:** gosales
 - Description:**

Again, the Lineage window appears and you can investigate the metadata on either the Business View tab or Technical View tab.

- Click the **Technical View** tab.

The results appear as follows:



Because you chose to view lineage on a query subject rather than a query item, you are presented with all the query items that make up the query subject, both hidden and visible items. You can click on any item to view more information about it in the Property pane.

- Close all browser windows and then close Framework Manager.

Results:

By using the Lineage feature in Cognos Viewer and the metadata tree in Query Studio, you were able to gain additional information about the data used in reports.

Create a Model Report

gosales	
Last Changed: 2007-11-13T14:56:44	
PRODUCT_TYPE <--> SALES_TARGET	
Status	valid
Name ()	PRODUCT_TYPE <--> SALES_TARGET
Expression	[gosales].[PRODUCT_TYPE].[PRODUCT_TYPE_CODE] [PRODUCT_TYPE_CODE]
Left	Refobj [gosales].[PRODUCT_TYPE]
	Mincard one
	Maxcard one
Right	Refobj [gosales].[SALES_TARGET]
	Mincard one
	Maxcard many

Select the entire model or any object within it, and then click Tools > Model Report. The details of every model object are presented. They can be saved in HTML or XML format for documentation purposes or for debugging.

To make searching easier, we recommend that you make a separate Model Report for each high-level namespace and/or folder in your project.

The above example shows the details of a relationship within the Foundation Objects View.

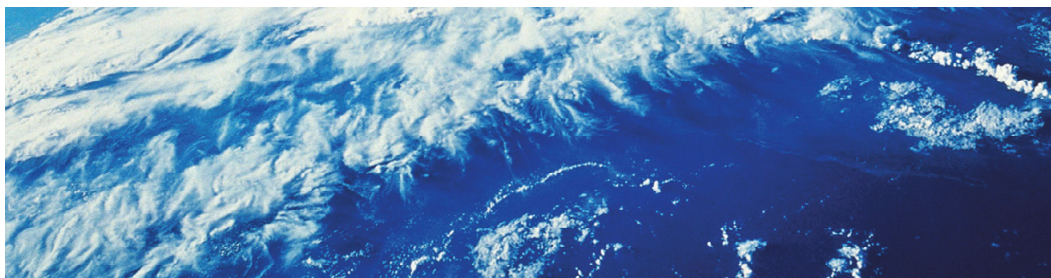
Summary

- You should now be able to:
 - perform basic maintenance and management on a model
 - remap metadata to another source
 - import and link a second data source
 - run scripts to automate or update a model
 - view lineage
 - create a model report



Optimize and Tune Framework Manager Models

IBM Cognos BI



Business Analytics

© 2010 IBM Corporation

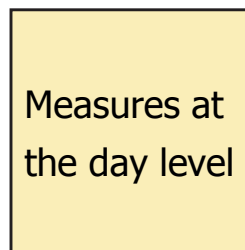
Objectives

- At the end of this module, you should be able to:
 - identify and implement techniques to optimize and tune your Framework Manager models

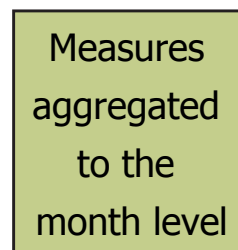
Materialized Views

- IBM Cognos queries typically aggregate data.
- Materialized views that contain aggregated values, can significantly improve performance.
- Database optimization can take advantage of materialized views.

Base Sales Fact Table



Materialized View
for Sales Facts



Cognos.
software

© 2010 IBM Corporation

Materialization is a benefit for relational models. However it is highly recommended with dimensional modeling. Modeling relational data dimensionally adds another layer in your model which enables OLAP behaviors in reports. To provide the best performance, materialization in the database should be in place.

Common database vendor implementations include:

- Microsoft SQL Server - view index
- Oracle - materialized view
- IBM DB2 UDB - materialized query tables
- NCR Teradata - aggregate join indexes
- Informix RedBrick - VISTA

Achieve Minimized SQL

- Minimized SQL may not occur when:
 - relationships are added to model query subjects
 - data source query subject SQL is altered
 - calculations, filters, or determinants are added
- SQL minimization should be considered as you model, not towards the end of development.

There may be times when losing SQL minimization is necessary. For example, you may require model query subjects with overriding relationships to control query execution paths. You may need to change the SQL on data source query subjects to meet specific application needs. You may also require filters, calculations, or determinants to affect the SQL generation.

Minimized SQL will have a better chance of taking advantage of database optimization compared to more complex SQL.

Set Limits on Query Execution Using Governors

- Reduce system resource requirements and improve performance by setting governors.
- You can restrict:
 - number of report tables
 - number of rows returned
 - query execution time
 - and so on

SQL is generated automatically on demand when you:

- run a report in an IBM Cognos studio
- test a query subject or relationship in Framework Manager
- create a new query subject by transforming query subjects or merging objects

By setting governors, you can restrict the number of tables retrieved by a query and restrict the number of rows returned. You can also set time limits for query execution, and restrict character length on binary large objects (BLOBs).

You can also set runtime activities, such as deny outer joins and cross-product joins, which may produce very large and resource-intensive queries.

Reuse Data When Running Reports

- Allow reports to run against cached data
- Report does not have to re-query the database
- Enable by setting governor

Do your business users require that they always see the latest data? If not, take advantage of query re-use.

When you run a report for the first time, the query request is sent to the database and the result set is returned.

The query and result are stored in the cache for each user for their current session.

When you run the report again, it can often be created using the cached query and result set, without querying the database again.

To take advantage of caching for reports, you can enable the Allow Usage of Local Cache governor on the model in Framework Manager.

For those users that must see the latest data, consider publishing a separate package without the governor turned on. In this scenario, some users can leverage re-use while others cannot.

Demo 1: Set Limits on Query Execution Using Governors

Purpose:

The Great Outdoors IT department has requested that some query limits be set on the model to reduce the load on their servers. You will limit the number of tables any one query can return, set a time-out period for queries, and limit the number of rows retrieved by setting the appropriate governors.

Components: Framework Manager, Business Insight Advanced

Project: great_outdoors_warehouse

Task 1. Set the Report table limits governor.

1. In **Framework Manager**, close any projects that may be open, and then open the **great_outdoors_warehouse** project located at **C:\Edcognos\B5152\CBIFM-Start Files\Module 21\great_outdoors_warehouse**.
2. Save the project as **C:\Edcognos\B5152\Course_Project\great_outdoors_warehouse2**.
3. From the **Project** menu, click **Edit Governors**.
4. Change the value in the **Maximum number of report tables** box from **0** to **3**, and then click **OK**.
5. Save the project.

Task 2. Test your Report table limits governor setting.

1. Publish the **GO Data Warehouse (query)** package.
Tip: Clear the **Verify the package before publishing** check box to save time.
2. Launch **IBM Cognos Connection**, login, and then launch **Business Insight Advanced** selecting the **GO Data Warehouse (query)** package for a **List** report.

3. Expand **Sales and Marketing (query)**>**Sales (query)**, and then drag the **Product** query subject to the work area.

An error message appears. The error occurs because the query subject generates SQL which references more than the allowable number of tables as defined by the governor setting. You will need to adjust it.

4. Click **OK**, and then close **Business Insight Advanced** and **IBM Cognos Connection**.

You are closing IBM Cognos Connection to clear the session cache before testing your new governor setting in the next step.

5. In **Framework Manager**, adjust the **Maximum number of report tables** governor value from **3** to **20** and save the project.
6. Re-publish the **GO Data Warehouse (query)** package.
7. Re-test the same **Product** query subject in **Business Insight Advanced**.
The item can be successfully added to the report.
8. Close **Business Insight Advanced** and **IBM Cognos Connection**.

Task 3. Set the Query execution time limits governor.

1. In **Framework Manager**, in the **Governors** dialog box, change the **Query execution time limit (seconds)** value from **0** to **1**, and then click **OK**.
2. Save the project.

Task 4. Test your Query execution time limits setting.

1. Re-publish the **GO Data Warehouse (query)** package.
2. Launch **IBM Cognos Connection**, launch **Business Insight Advanced**, and then select the **GO Data Warehouse (query)** package for a **List** report.
3. Expand **Sales and Marketing (query)**>**Sales target (query)** and **Sales (query)**.

4. Select and drag several query items from different query subjects including **Sales fact** and **Sales target fact** to the work area at the same time.

An error message appears indicating query execution time exceeds the 1 second limit. This governor can be adjusted as needed. For now, you will reset it to the default value of 0 which indicates an unlimited time limit.

5. Close **IBM Cognos Connection**.
6. In **Framework Manager**, set the **Query execution time limit (seconds)** governor value back to **0**.
7. Leave the **Governor** dialog box open for the next task.

Task 5. Set and Test the Data retrieval limits governor.

1. Change the **Maximum number of retrieved rows** value from **0** to **25**, and then click **OK**.
2. Save the project, and then re-publish the **GO Data Warehouse (query)** package.
3. Launch **IBM Cognos Connection** and **Business Insight Advanced** selecting the **GO Data Warehouse (query)** package for a **List** report.
4. Expand **Sales and Marketing (query)**>**Sales (query)**>**Product**, and then drag **Product name** to the work area.

A message appears indicating that the number of rows retrieved exceeds the limit of 25.

5. Close **IBM Cognos Connection**.
6. In **Framework Manager**, in the **Governors** dialog box, change the **Maximum number of retrieved rows** value from **25** to **0**.
7. Click **OK**, and then save the project.

Results:

By setting, testing, and refining our governor settings, you ensured that the IT department's servers will not be unnecessarily overloaded.

Run Time Model (RTM) File

- The RTM file:
 - contains the package metadata
 - is the local cache for the server to retrieve metadata
 - is created on the server when a user first accesses a package in any studio
 - is stored in <IBM Cognos install location>\data\cqe\RTModels
 - increases in size depending on how much is published in the package

Only include the necessary metadata in the packages you publish for user consumption.

The size of the RTM file directly impacts the open time for the package in the studios. The larger the file size the longer it will take to open.

File size can be especially significant for packages that contain cube data sources, because the RTM files for these packages will be similar in size to the cubes.

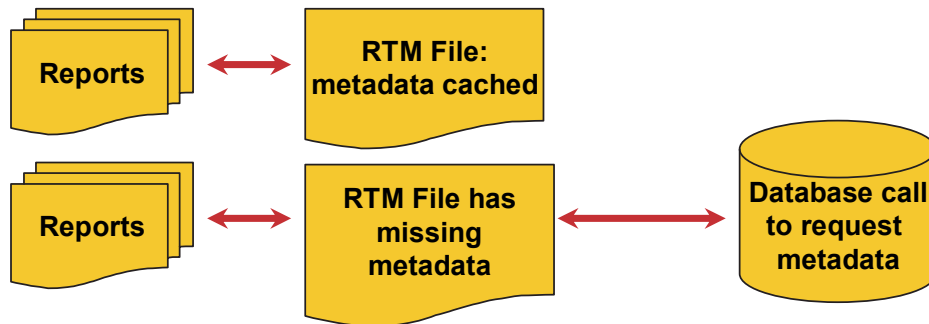
Examine Metadata Caching

- Framework Manager metadata:
 - describes objects imported from your data source and any objects created from them
 - is cached in the RTM file

By default, queries will use the cached RTM file metadata.

Enabling the "Allow Enhanced Model Portability" option in the project governors forces the fetching of metadata from the database and not from the model. Unless the model is being used in multiple environments where the columns in the database could potentially change in data type, disable this setting during production use of the model. There are cases such as during a model upgrade from ReportNet to IBM Cognos where the metadata should not be preserved so that a subsequent resynchronization of the model can appropriately update the data types of the modeled data items.

How do Reports Use Cached Metadata?



- Reports require metadata and descriptions of that metadata to fulfill queries.
- If any metadata is not cached in the RTM file, the result is additional database calls.

Query performance may become an issue if there are numerous call backs for metadata.

Once the data source connection has the metadata information, it is cached in memory for the duration of the connection only. The duration of the connection depends on many factors.

When is Metadata Not Cached?

- Metadata is not cached when you:
 - enable the model portability governor
 - modify a data source query subject by:
 - editing the SQL statement
 - adding embedded filter or calculation objects

As the need for IBM Cognos to interactively query the underlying relational database for metadata increases, there is a reciprocal decrease in performance.

Demo 2: Identify the Use of Cached Metadata by a Query

Purpose:

With the goal of identifying how the modification of data source query subjects can affect query performance, you will identify when the queries generated from our data source query subjects are using cached metadata, and when they are not.

Component: **Framework Manager**

Project: **great_outdoors_warehouse2**

Task 1. View a metadata call back to the database in Framework Manager.

You will first add a calculation to a data source query subjects and then view the effects of that calculation.

1. In the **Project Viewer**, expand **go_data_warehouse>Database view>Sales and marketing data**, and then double-click **SLS_SALES_FACT**.
2. Click the **Calculations** tab, then click **Add**.
3. Create the following calculation called **Planned revenue**:

**[Sales and marketing data].[SLS_SALES_FACT].[QUANTITY] *
[Sales and marketing data].[SLS_SALES_FACT].[UNIT_PRICE]**

Now this data source query subject includes an embedded calculation.

4. Click **OK** twice, and then test any query item from the **SLS_SALES_FACT** query subject.

5. Click the **Query Information** tab, and then click the **Response** link at the top of the dialog box.

The xml response from the IBM Cognos server for this query appears.

Notice the text shown below:

RQP-DEF-0543 Metadata will be retrieved from the database, because a simple DB query subject definition with matching metadata does not exist.

Whenever this message is encountered in your testing, you will know that the data source query subject has been altered and is no longer considered a simple query subject. In this case it occurs because the query subject contains a calculation which results in a dynamic query, which cannot be resolved using the metadata currently cached in the model.

Note: This test can also be run in Report Studio by going to the Tools menu, Validate Options, and selecting Information in the Validation level list. Now when you validate the report, you will be able to see this message if it is generated.

6. Click **Close**, and then remove the calculation from the **SLS_SALES_FACT** query subject and test again.

The metadata call back message no longer appears since cached metadata is used.

7. Save the project.

Results:

You identified when the queries generated from our data source query subjects use cached metadata in the model, and when they do not. This was achieved by viewing the query response information.

Control Where Queries are Processed

- To improve performance, control where the query will be processed.
- There are two types of query processing:
 - limited local
 - database only

For relational metadata, you can improve performance by selecting the appropriate type of query processing for the data source. You can specify whether SQL processing is performed by the database server or processed locally. This setting will depend on your database and IBM Cognos servers' processing power and other factors found in the documentation.

Control Computation of Aggregates

- Computation of aggregate values can be process-intensive.
- To improve performance, specify where aggregates are computed (at the database or application level).
- There are four settings for rollup processing:
 - Default/unspecified
 - Extended
 - Database
 - Local

The default setting for rollup processing is "default" in Report Studio and "unspecified" in Framework Manager, but can be set to Extended, Database, or Local.

Demo 3: Examine Rollup Processing and Generated SQL

Purpose:


A report author was interested in allowing IBM Cognos to perform certain aggregations locally rather than at the database level. You will show the report author how to accomplish this in Report Studio and what to expect in the generated SQL.

Components: Report Studio, Framework Manager

Project: great_outdoors_warehouse

Package: GO Data Warehouse (query)

Task 1. Author a report with summary values.

1. In **IBM Cognos Connection**, launch **Report Studio** selecting the **GO Data Warehouse (query)** package.
2. Create a new **List** report, and then in the **Insertable Objects** pane, expand the **Sales and Marketing (query)>Sales (query)>Retailer**.
3. Drag the **Retailer name** query item to the report.
4. From **Sales Fact**, add **Quantity** to the report.
5. Click the **Quantity** column, and then on the toolbar click **Aggregate** , and then click **Total**.

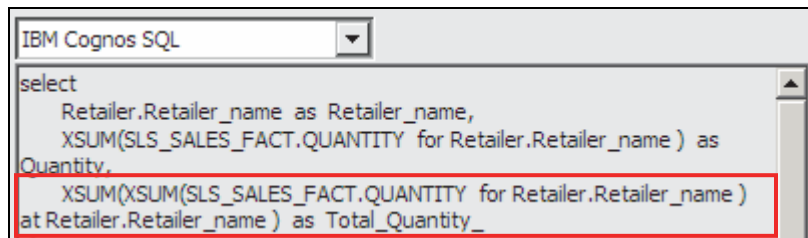
The results appear as follows:

Retailer name	Quantity
<Retailer name>	<Quantity>
<Retailer name>	<Quantity>
<Retailer name>	<Quantity>
Overall - Total	<Total(Quantity)>


Task 2. View the Generated SQL and set the appropriate option.

1. From the **Tools** menu, click the **Show Generated SQL/MDX**.
2. In the list, click **IBM Cognos SQL**.

The SQL appears as shown below:

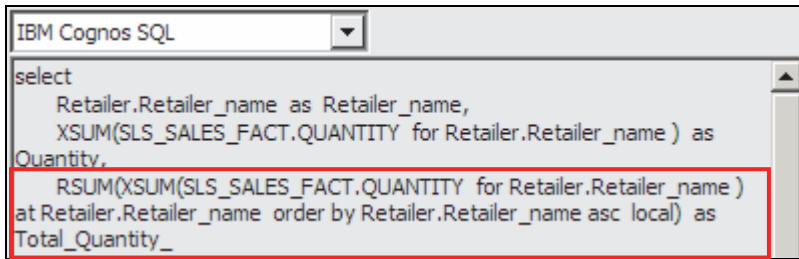


Notice that Total_Quantity is totaled for Retailer.Retailer_name. This represents the summary total at the bottom of the report. This aggregated total is displayed using extended aggregates based on the presence of the XSUM(XSUM) function. This SQL was generated based on the default setting. With this setting, you always see SQL generated based on batch mode.

3. Click **Close**.
4. On the **Explorer** bar, point to **Query Explorer** , and then click **Query 1**.
5. In the **Properties** pane, under **Query Hints**, click the **Rollup Processing** row, and then in the list, click **Local**.

6. View the generated **IBM Cognos SQL** again.

The SQL appears as shown below:



```

IBM Cognos SQL
select
  Retailer.Retailer_name as Retailer_name,
  XSUM(SLS_SALES_FACT.QUANTITY for Retailer.Retailer_name ) as
  Quantity,
  RSUM(XSUM(SLS_SALES_FACT.QUANTITY for Retailer.Retailer_name )
  at Retailer.Retailer_name order by Retailer.Retailer_name asc local) as
  Total_Quantity_

```

The aggregated total is now displayed as using running aggregates based on the presence of the RSUM function. This setting ensures local processing of the summary totals no matter what the output format of the report is.

7. Click **Close**.
8. Change the **Rollup Processing** property to **Extended**, and view the generated IBM Cognos SQL.

The SQL appears the same as in the first example because you are explicitly asking for an extended aggregate.

9. Change the **Rollup Processing** property to **Database**, and view the generated IBM Cognos SQL.

Again, the RSUM function is present. The Database setting instructs the IBM Cognos query engine to use running aggregates where possible. The Database setting pushes the running aggregation to the database using derived tables if the database vendor supports this feature. If not, the running aggregation is performed locally.

10. Close **Report Studio** without saving.

Task 3. Examine the Rollup Processing property in Framework Manager.

1. In **Framework Manager**, in the **Project Viewer**, expand **Data Sources**, and then click **go_data_warehouse**.

In the Properties pane, the value for the Rollup Processing property is set to unspecified, rather than a default option. These are the same. Setting Extended, Database or Local in the model affects the default behavior for every report that is created from it, when the corresponding property in Report Studio is set to Default. As seen earlier, you can override the model setting for an individual report in Report Studio.

2. Leave IBM Cognos Connection, and Framework Manager open for the next demo.

Results:

You demonstrated how to force local processing of summary totals by setting the Rollup Processing property in Report Studio.

Use Filters to Improve Performance

- Reducing rows retrieved can improve performance.
- You can use:
 - embedded filters
 - embedded filters with prompts
 - encourages users to focus their queries
 - stand-alone filters
 - security filters

You can build mandatory filters into the model to ensure that consumers do not retrieve excessively large data sets when running reports.

Is it a requirement that users see all the data all of the time? If not, consider adding mandatory filters to the model.

All filter types (listed above) limit the data set retrieved, resulting in a decreased query processing load.

You should consider filtering data sets when modeling relational data dimensionally. This will help offset any decrease in performance from having an additional layer of metadata in your model.

Apply Design Mode Filters

- Use design mode filters to improve performance when:
 - testing query subjects in Framework Manager
 - designing reports in Report Studio and Query Studio

Apply design mode filters in query subjects to limit the amount of data that report authors and modelers retrieve when testing and designing.

By limiting data retrieval, design time results appear more quickly.

Demo 4: Apply Design Mode Filters

Purpose:

You want to control how long it takes for queries to run when report authors are designing reports in Query Studio and Report Studio. To do this you will begin by adding a design mode filter to one of the query subjects in our model.

Components: Framework Manager, Query Studio

Project: great_outdoors_warehouse

Package: GO Data Warehouse (query)

Task 1. Create a design mode filter.

1. In **Framework Manager**, in the **Project Viewer**, expand **go_data_warehouse>Business** view.
2. Open the **Query Subject Definition** dialog for **Sales fact**, and then click the **Filters** tab.
3. Create a filter called **Limited Products Design Mode Filter** with the following expression:

[Sales and marketing data].[SLS_SALES_FACT].[PRODUCT_KEY] in (30001, 30002, 30003, 30004, 30005)

Hint: The Sales and marketing data namespace is found in the Database view namespace.

4. Click **OK** to close the **Filter Definition** dialog box.

Notice that the current Usage for the filter is set to Always.

5. Under the **Usage** column, click the **ellipsis**.
6. In the list, click **Design Mode Only**, and then click the **Test** tab.
7. Click the **Options** link in the lower right corner, select the **Apply all relevant design mode filters when testing**, and then click **OK**.
8. Click **Test Sample**.

Test results (Design Mode filter applied)							
	Quantity	Unit cost	Unit price	Unit sale price	Gross margin	Revenue	Gross
	1172	6.62	12.53	8.77	0.2452	10278.44	2519.8
	591	34.97	54.93	52.18	0.3298	30838.38	10171.11
	2649	2.9	6.59	6.13	0.5269	16238.37	8556.27
	2094	2.9	6.59	6.19	0.5315	12961.86	6889.26
	991	0.85	3.66	3.55	0.7606	3518.05	2675.7
	467	34.97	54.93	52.73	0.3368	24624.91	8293.92
	934	6.62	12.53	8.77	0.2452	8191.18	2008.1
	715	15.93	23.8	21.42	0.2563	15315.3	3925.35

Notice that the test results have the design mode filter applied. The values retrieved are restricted to the product key specified in the filter.

9. Open the **Options** dialog box, clear the **Apply all relevant design mode filters when testing**, click **OK**, and then click **OK** again to close the **Query Subject Definition** dialog box.

Task 2. Test the design mode filter in Query Studio.

1. Publish the **GO Data Warehouse (query)** package and then save the project.
2. In **IBM Cognos Connection**, launch **Query Studio**, and then select the **GO Data Warehouse (query)** package for a query.

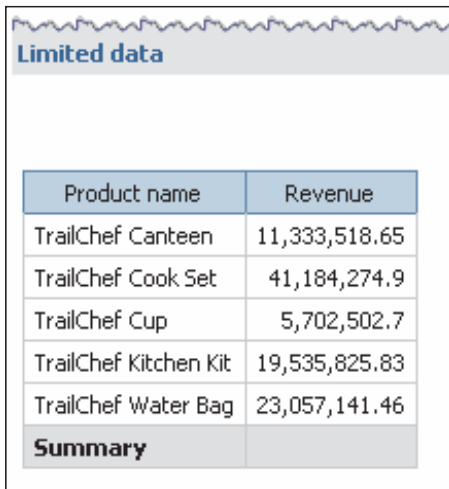
- Expand **Sales and Marketing (query)>Sales (query)**, and then add the following items to the report:

Query Subject	Query Item
Product	Product name
Sales fact	Revenue

A list report displaying pages of all products and their revenue appears.

- Under **Menu**, click **Run Report**, and then click **Preview with Limited Data**.

The report appears as shown below:



Limited data	
Product name	Revenue
TrailChef Canteen	11,333,518.65
TrailChef Cook Set	41,184,274.9
TrailChef Cup	5,702,502.7
TrailChef Kitchen Kit	19,535,825.83
TrailChef Water Bag	23,057,141.46
Summary	

The design mode filter is applied and the Sales fact Revenue values are restricted to only the products specified in the filter.

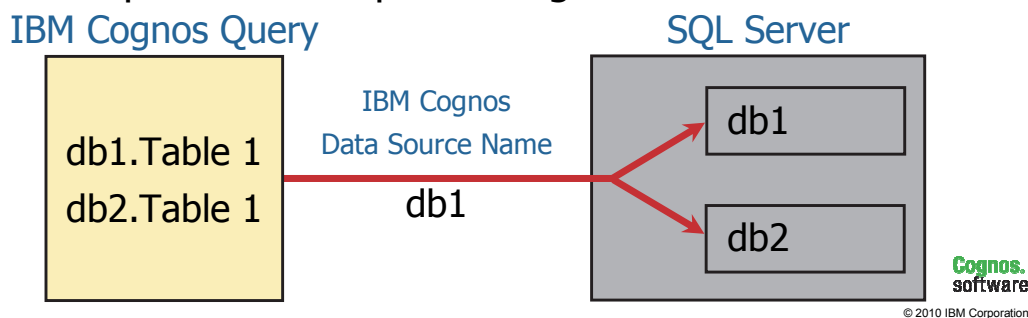
- Return to **IBM Cognos Connection** without saving your query.

Results:

You added a design mode filter to one of the query subjects in our model. Doing this lets you control how long it takes for queries to run when a report author is designing a report that uses this query subject.

Reduce Database Connections

- Use the same IBM Cognos data source connection name:
 - for all data source connections to the same database server instance
 - to reduce database connections
 - to prevent local processing



Using the same data source name for multiple data source connections to the same database instance allows for fewer database connections at run time and allows the database server to perform the joins between the databases rather than the IBM Cognos servers.

This technique can be implemented in the data source properties in Framework Manager.

Indicate the Performance Impact of Functions

- Provide visual clues about the performance of functions
- Can be set at the project or package level

Quality of Service Indicators

	Not Available
	Limited Availability
	Poor Performance
	Unconstrained

Through a Framework Manager model, report authors can write reports that query any combination of data source types, but not all data sources support functions the same way.

The quality of service indicator provides report authors with a visual clue about the performance of individual functions when used in conjunction with the data sources accessed by the model. This lets report authors avoid using functions that could result in long running queries or queries that fail.

You can also provide descriptive text about a function.

Demo 5: Indicate the Performance Impact of Functions

Purpose:


Report authors must be aware of the impact of using certain functions on the performance of their reports. To help them identify potential performance issues with using functions, you will add a quality of service indicator to a function that potentially may cause performance issues. You will then publish the function (as part of a set) with a package.

Components: Framework Manager, Report Studio

Project: great_outdoors_warehouse

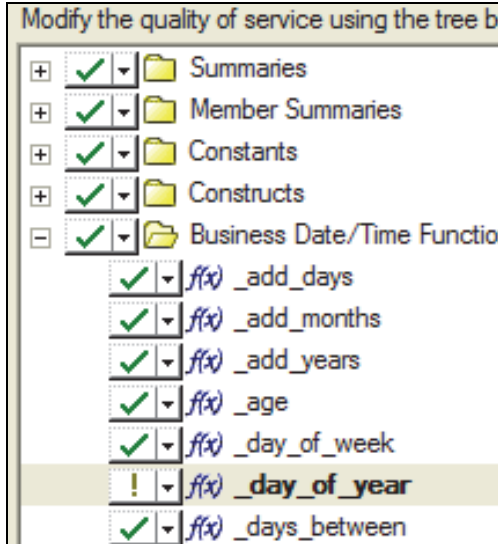
Package: GO Data Warehouse (query)

Task 1. Define quality of service indicator for a function.

1. In the **Project Viewer**, click the **GO Data Warehouse (query)** package.
2. From the **Actions** menu, point to **Package**, and then click **Specify Package Function List**.
3. Under **Selected function sets**, Ctrl+click all items except **DB2**, and then click **Remove** .
4. Click **Define Quality of Service**.

- Expand the **Business Date/Time Functions** folder, from the indicator list, next to the **$f(x)$ _day_of_year** function, click the down arrow, and then click **Limited Support**.

The results appear as follows:




- Click in the box next to the list of functions, and then type **This function may adversely affect performance if used against non-filtered data sets**.

This information becomes available to your report authors and can assist them in determining whether to use this function in their reports.


- Click **OK**, and then click **OK** again.
- Save the project.

Task 2. Publish the package and test the quality of service indicator.

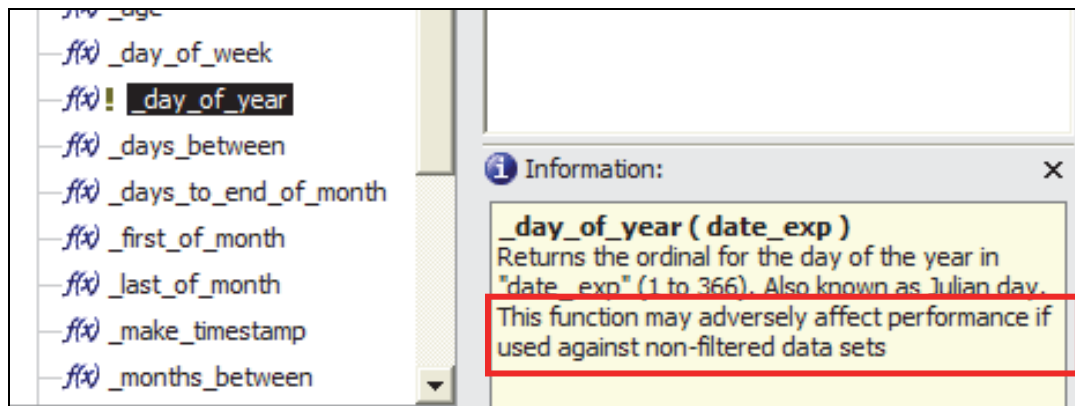
- Publish the **GO Data Warehouse (query)** package.

2. In **IBM Cognos Connection**, launch **Report Studio** selecting the **GO Data Warehouse (query)** package, and then create a **List** report.
3. Expand **Sales and Marketing (query)>Sales (query)>Time dimension**.
4. Drag the **Date** item to the report.
5. In the report, double-click the **Date** column.
6. Under the **Available Components** pane, click the **Functions**  tab.
7. Expand the **Business Date/Time Functions** folder.

Notice the Limited Support  indicator within the *f(x)_day_of_year* function.

8. Click the **f(x) _day_of_year** function.

The results appear as follows:



In the Information pane, notice the text that you added as additional information about the function earlier.

9. Click **Cancel**, and then close **Report Studio** without saving.
10. Close **Internet Explorer**, and then, in **Framework Manager**, save and close your project.

Results:

You added a quality of service indicator to a function and published a package that contains it. Report authors will be aware of the impact of using this function on the performance of their reports.

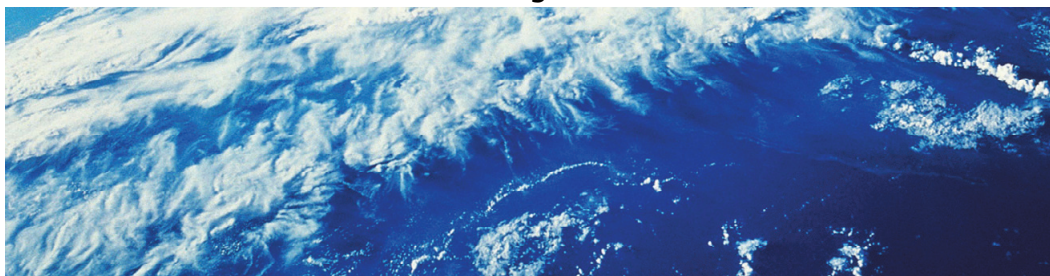
Summary

- You should now be able to:
 - Identify and implement techniques to optimize and tune your Framework Manager models



Work in a Multi-Modeler Environment

IBM Cognos BI



Business Analytics

© 2010 IBM Corporation

Objectives

- At the end of this module, you should be able to:
 - implement repository control
 - segment and link a project
 - branch a project and merge results

Use Repository Control

- Use a repository control system of your choice on Framework Manager project files to:
 - manage and backup versions of you project
 - prevent change conflict between users by checking files in and out
 - ensure modelers are always working on the latest version

You can manage Framework Manager projects with an external source control system. Please refer to the documentation on how to put the Framework Manager project files into an external repository.

Framework Manager Projects consist of three essential files, plus a log file directory. These files are:

File Name	Optional/Required	Description
[Project name].cpf	Required	This project file tracks segments and links.
model.xml	Required	This is the model data.
customdata.xml	Required	This saves user interface information such as diagram layout.
preferences.xml	Optional	This file is not used.
repository.xml	This file must not be present if you are handling your own repository. If this file is in your project, you must delete it.	

Name	Size	Type	Date Modified	Att
logs		File Folder	5/9/2005 4:37 PM	
customdata.xml	90 KB	XML Document	10/4/2004 2:57 PM	A
go_data_warehouse.cpf	2 KB	BMT Project File	5/9/2005 3:59 PM	A
model.xml	2,014 KB	XML Document	5/9/2005 3:59 PM	A
Preferences.xml	1 KB	XML Document	6/7/2004 11:38 AM	A

Log files are found in the "logs" subdirectory. A new log file is created each for each Framework Manager session. These logs are required only if you intend to use the Model Synchronization feature. External repository handling is simpler if you ignore the log files.

Name	Size	Type	Date Modified	Att
go_data_warehouse-20041005105202-log.xml	18 KB	XML Document	10/5/2004 11:01 AM	A
go_data_warehouse-20041005111445-log.xml	2 KB	XML Document	10/5/2004 11:16 AM	A
go_data_warehouse-20050509155417-log.xml	1 KB	XML Document	5/9/2005 3:59 PM	A
go_data_warehouse-20050509162817-log.xml	1 KB	XML Document	5/9/2005 4:28 PM	A
go_data_warehouse-20050509163719-log.xml	1 KB	XML Document	5/9/2005 4:37 PM	A
go_data_warehouse-20050509164739-log.xml	1 KB	XML Document	5/9/2005 4:47 PM	A

1. After your Framework Manager project is created, make sure it is closed, and then, 'Check In' the project file.cpf, model.xml, and customdata.xml into your repository of choice.
2. Before opening the Framework Manager project to begin modeling again, you will be required to 'Check Out' those three files out from your repository.

NOTE: If you fail to do so (and your repository system marks checked in files as read-only), Framework Manager will open in read-only mode. The text "[Read Only]" will appear on the application title.

SOLUTION: If this happens, close the project, and check out the files. If you have done a lot of work in a read-only project, then you may save the project, and you must then ensure those files get checked in properly.

3. When your Framework Manager session is complete, close the project, and then 'Check In' these three files to your repository.

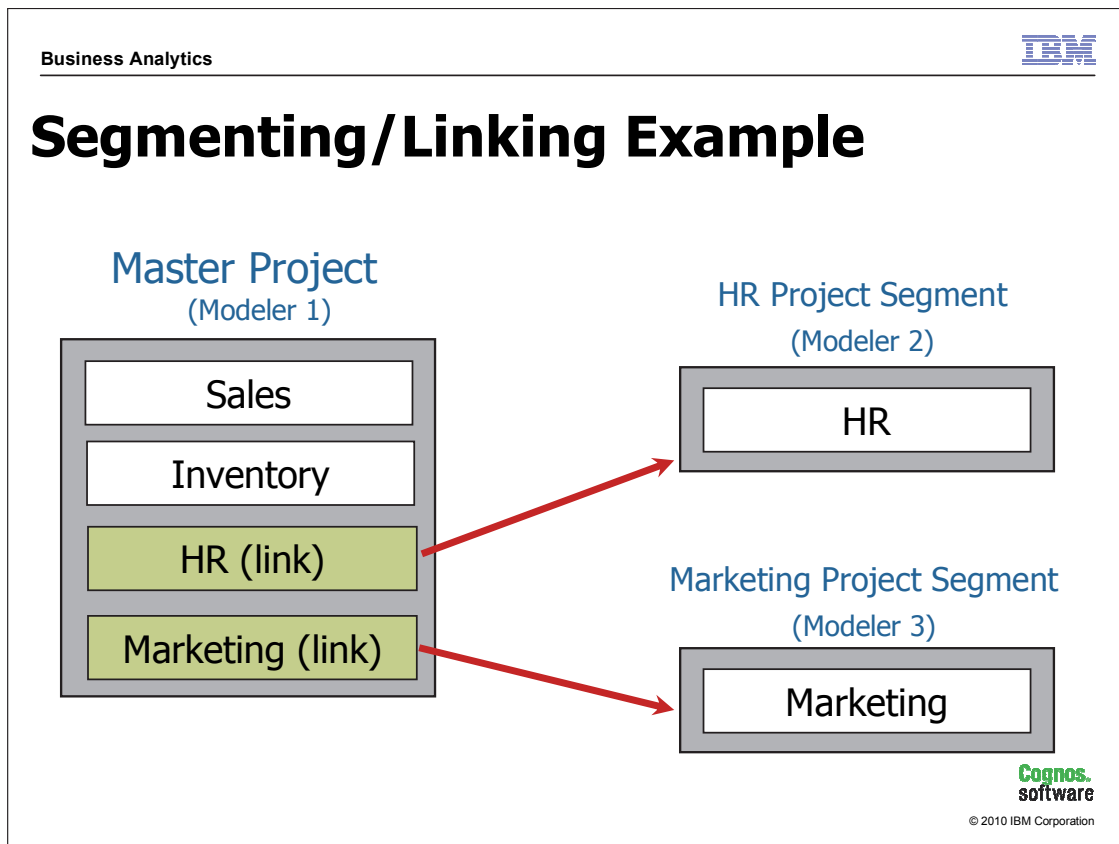
Create a Segment

- Creates a new project with its own project files
- Use segments to:
 - distribute a project according to business rules and organizational requirements
 - share and re-use project information with other projects

When you create a segment, you create a new project in a new folder, complete with its own associated project files. The new project is linked by a shortcut in the main project from which it was created.

You can only segment a project either at the folder or namespace level.

The master project has access to the entire model, including the segments.



Modeler 1 can create segments for the HR and Marketing metadata. These segments are new projects created by Framework Manager and become links in the master project.

Modeler 2 will work on the HR metadata, to model for predictable results, while modeler 3 will work on the Marketing metadata. The changes they make will be reflected in the master project.

Communication is required between the modelers to ensure that no one overwrites each others changes. For example, modeler 1 should not make any changes to the HR or Marketing metadata without talking to the respective segment owner otherwise modeling conflicts might occur and changes may be lost.

The HR and Marketing segments can also be linked into other projects that may require this type of information.

Regarding repository control:

For segmented projects, the segments are simply project directories stored under the parent project directory. There are two ways to work with a segmented project.

1. The segments can be individually opened as stand-alone projects, in which case repository handling is the same as for any other project.
2. Or segments can be opened as part of the main project. In this case you have to 'Check Out' the projects for each segment you intend to modify, which are located, as sub-directories under the main project.

NOTE: The repository should maintain the same hierarchy as the project directory.

If you do require the Model Synchronization feature, you must 'Check In' any new log files that are created. When you are ready to synchronize, you will need to get a copy of **all** the project log files.

NOTE: It is not necessary to 'Check Out' the log files, as they are never updated after they are first created.

Demo 1: Create a Segment

Purpose:

You want to distribute the workload of a project to another modeler. You want the other modeler to work on employee related data, while you work on other business areas.

Component: **Framework Manager**

Project: **New Project**

Task 1. Create a project.

1. In **Framework Manager**, close any open projects, and then create a new project.
2. In the **Location** box, navigate to **C:\Edcognos\B5152\Course_Project**.
3. In the **Project name** box, type **great_outdoors_warehouse_MASTER** and then click **OK**.
4. On the **Select a Language** dialog, ensure **English** is selected, and then click **OK**.
5. Ensure **Data Sources** is selected, and then click **Next**.
6. Click **great_outdoors_warehouse**, and then click **Next**.
7. Expand **GOSALESDW>Tables**, and then select all tables prefixed with **EMP** and **SLS**.

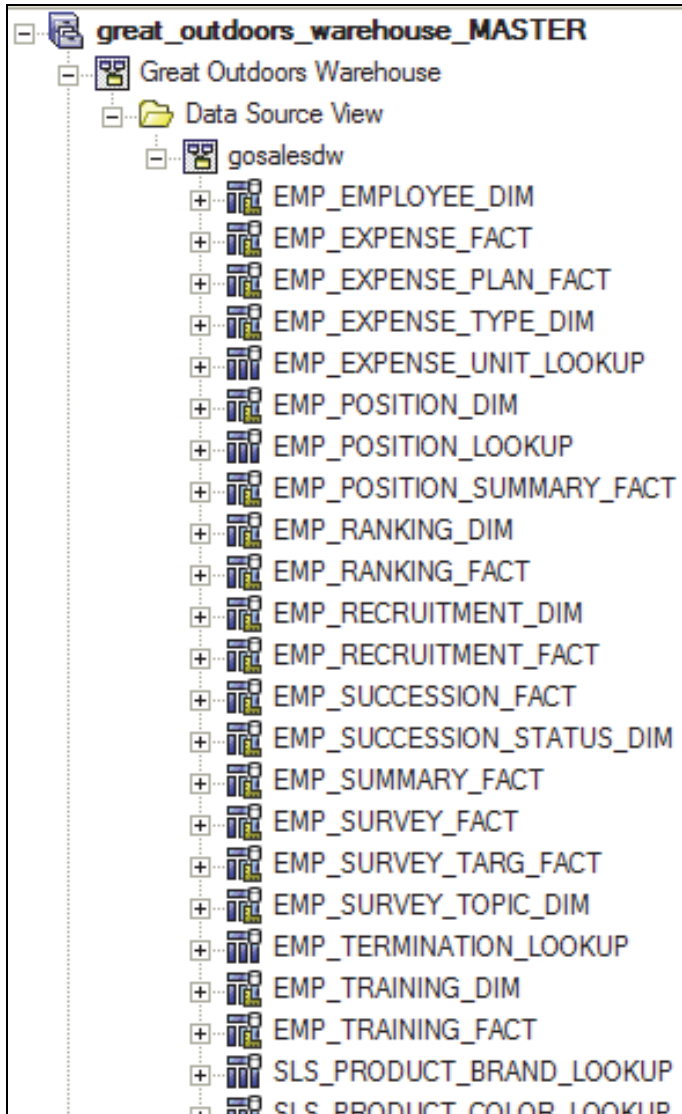
Tip: You can select the first EMP_ table, scroll down to the last EMP_ table, and then Shift+click the last EMP_ table to select them all. Then repeat for the SLS_ tables.

8. Click **Next**, **Import**, and then **Finish**.

You will rename the root namespace and organize the imported objects.

9. Rename the root namespace to **Great Outdoors Warehouse**.
10. Create an empty folder in the **Great Outdoors Warehouse** namespace called **Data Source View**.
11. Create a namespace in the **Data Source View** folder called **gosalesdw**.
12. In the **Great Outdoors Warehouse** root namespace, select all **query subjects** and drag them into the **gosalesdw** namespace.

The results appear as follows:



13. Save the project.

Task 2. Create a folder containing objects you would like to segment.

1. In the **Project Viewer**, in the **gosalesdw** namespace, select all data source query subjects prefixed with **EMP**.

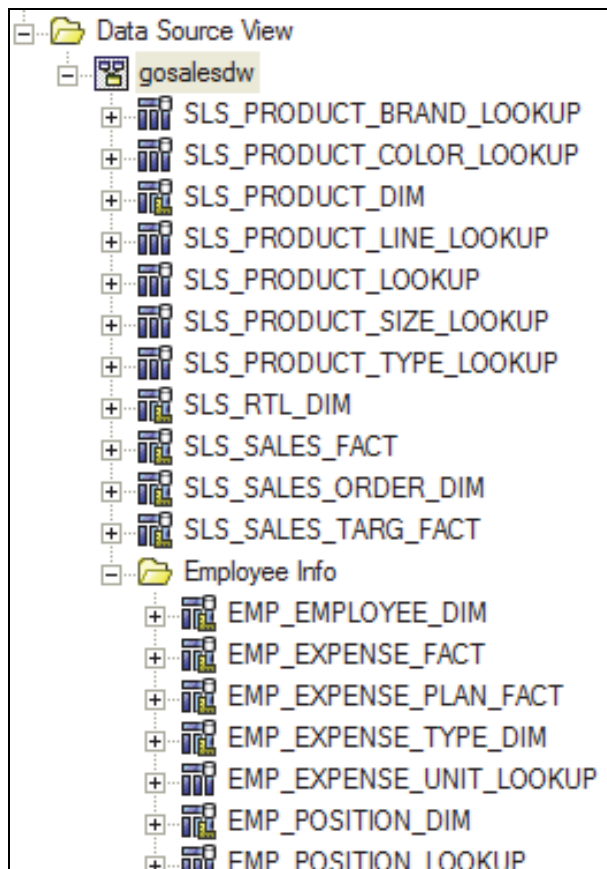
You will create a new parent folder for the selected objects.

2. Right-click one of the selected query subjects, point to **New Parent**, and then click **Folder**.

The employee data source query subjects are now contained in the new folder

3. Rename **New Folder** to **Employee Info**.
4. Expand the folder to view the objects inside.


The results appear as shown below:



5. Save the project.

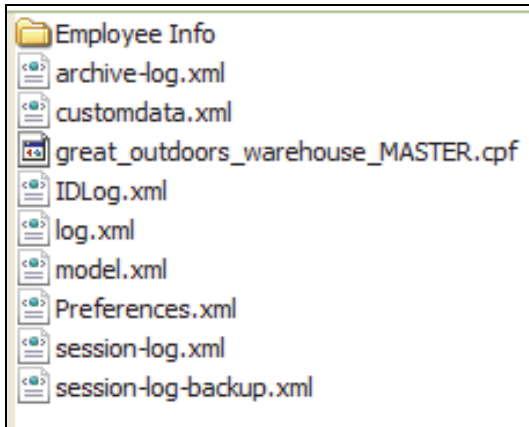
Task 3. Create a segment and view project files.

1. Right-click the **Employee Info** folder, and click **Create Segment**.
2. Accept the defaults and click **OK**.

The Employee Info folder is now represented by a link icon .

3. Open **Windows Explorer** and navigate to **C:\Edcognos\B5152\Course_Project\great_outdoors_warehouse_MASTER**.

The results appear as follows:



Notice the Employee Info folder.

4. Open the **Employee Info** folder.

Notice a new set of Framework Manager project files. Now the second modeler can work on this separate project and have the changes reflected in the main project.

Task 4. Make changes in the segmented project.

1. Return to **Framework Manager**, and then close the **great_outdoors_warehouse_MASTER** project and click **Yes** to save the changes.
2. Open the **Employee Info** project.
3. In the middle pane, click **Diagram**.

4. In the **Project Viewer**, expand **gosalesdw>Employee Info**.
5. Right-click **EMP_TERMINATION_LOOKUP**, and then click **Locate in Diagram**.
6. Modify cardinality of the relationship as follows:
EMP_TERMINATION_LOOKUP (TERMINATION_CODE, 1..1) to
EMP_EMPLOYEE_DIM (TERMINATION_CODE, 1..1)
7. Change the **Usage** property for **TERMINATION_CODE** in **EMP_EMPLOYEE_DIM** to **Attribute**.
8. Save and close the project.

Task 5. View changes in main project.

1. Open the **great_outdoors_warehouse_MASTER** project.
2. In the **Project Viewer**, locate and expand the **EMP_EMPLOYEE_DIM** query subject in the linked **Employee Info** segment.

Notice that **TERMINATION_CODE** is now set as an attribute.

3. Open **EMP_TERMINATION_LOOKUP** in the **Context Explorer** and select **Show Related Objects**.

Notice the relationship to **EMP_EMPLOYEE_DIM**. All the changes made in the segment project are reflected in the main project.

4. Close the **Context Explorer**, and then close the project.

Results:

You distributed the workload of a project to another modeler by creating a segment. Changes made to the segment were reflected in the main project.

Create a Link

- Create links to existing projects to:
 - organize work across large projects
 - maintain consistency
 - re-use information
- You can only create links to:
 - folders
 - namespaces
 - projects

A link is a shortcut to an existing project, folder, or namespace.

You must create the project, folder or namespace before you can link to it.

Demo 2: Create a Link

Purpose:

You will create a new project that will model the marketing metadata of the Great Outdoors Company. This project can also be leveraged in the `great_outdoors_warehouse_MASTER` project, so you will create a link to it.

Component: **Framework Manager**

Project: **New Project**

Task 1. Create a New Project for marketing information.

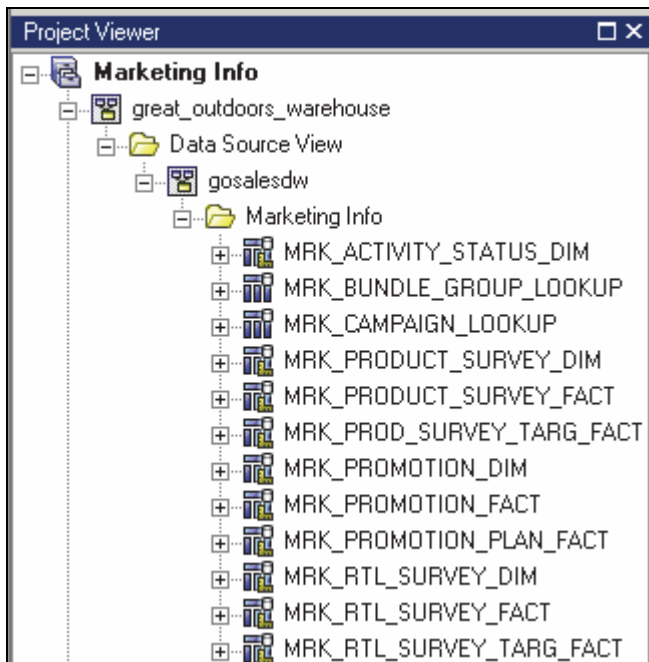
1. In **Framework Manager**, create a new project in the **Course_Project** folder.
2. Name the project **Marketing Info**, and then click **OK**.
3. If a message displays regarding the structure of the project, click **OK**.
4. Click **OK**, click **Next** twice, and then import all tables prefixed with **MRK** located in the **GOSALESDW** schema from the `great_outdoors_warehouse` data source.

You need to create the same namespace parent in this project as in the main `great_outdoors_warehouse_MASTER` project to link in the marketing metadata.

5. Rename the root namespace `great_outdoors_warehouse`.
6. In the **Project Viewer**, under the `great_outdoors_warehouse` namespace, create a folder called **Data Source View**, and then under that, create a namespace called `gosalesdw`.

- Under the **gosalesdw** namespace, create a folder called **Marketing Info**, and then move all the marketing query subjects into the new folder.

The results appear as shown below:



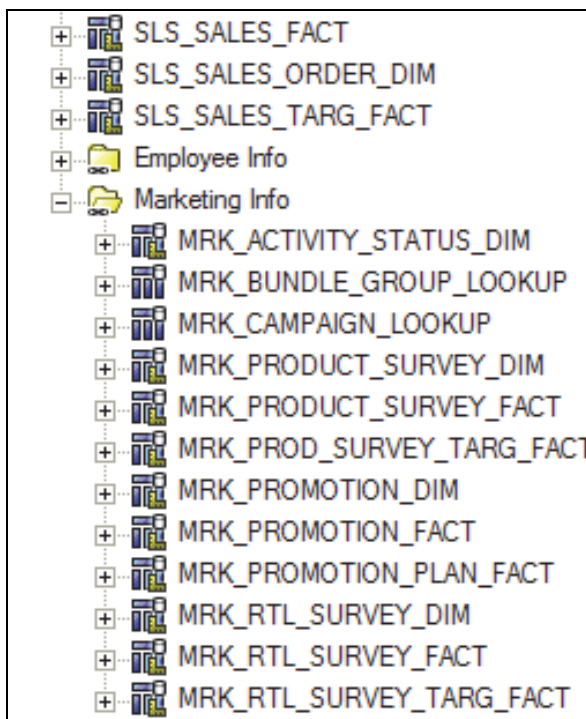
- Save and close the project.

Task 2. Create a link in the main project.

- Open the **great_outdoors_warehouse_MASTER** project.
- Right-click the **gosalesdw** namespace, and then click **Link Segment**.
- If a message appears regarding the structure of the project, click **Yes**.
- Browse to the **Marketing Info.cpf** file, and then click **Open**.
- Click **OK** to the UNC recommendation message, expand **great_outdoors_warehouse>Data Source View>gosalesdw**, and then click the **Marketing Info** folder.
- Click **Add**, and then click **OK**.

7. Expand **gosalesdw** and **Marketing Info**.

The Marketing Info project is now linked in the great_outdoors_warehouse_MASTER project as shown below:



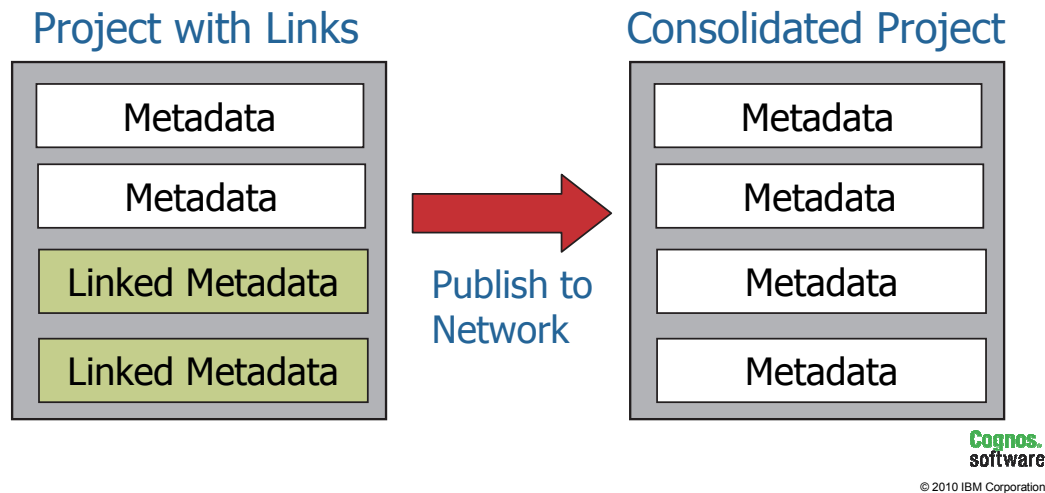
8. Save the project and leave it open for the next demo.

Results:

You successfully linked the Marketing Info project in the great_outdoors_warehouse_MASTER project.

Consolidate Linked Segments

- Consolidate linked segments in a project by publishing a package to a network location



The new package published to the network will include all the metadata specified by the modeler without any linked segments.

This process is useful when a multi-modeler environment is no longer required and you wish to consolidate your metadata into one project. This process is also useful if you would like assistance troubleshooting your model from support or other modelers. Rather than sending the master project and all the linked projects, you can send one consolidated model.

Demo 3: Consolidate Linked Segments

Purpose:

You would like to consolidate your multi-modeler environment by creating a single project that contains all the metadata without links to other projects.

Component: Framework Manager

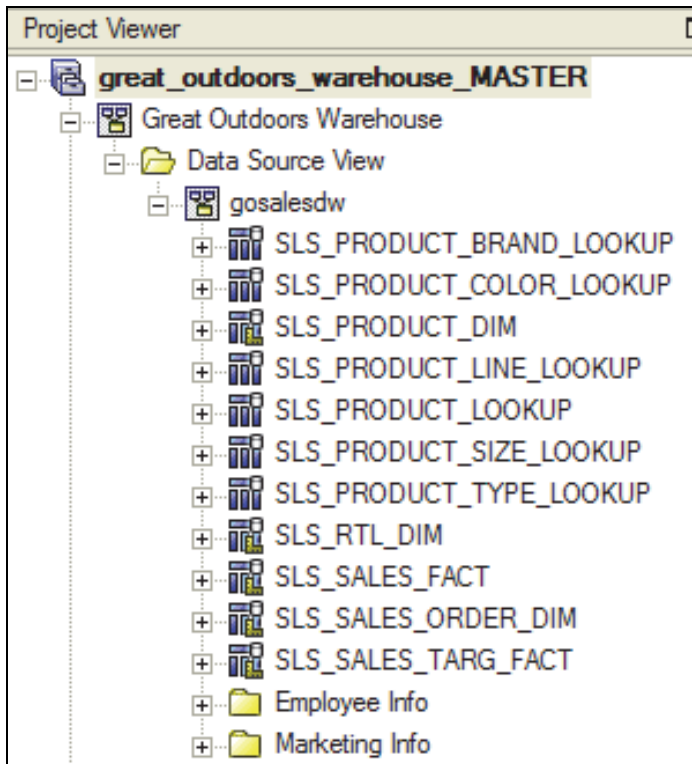
Project: great_outdoors_warehouse_MASTER

Task 1. Publish a package to the network.

1. In the **Project Viewer**, create a new package called **great_outdoors_warehouse_FINAL**.
2. Accept the defaults and click **Finish**.
3. Click **Yes** to start the **Publish Package Wizard**.
4. Select **Location on the network**, and then browse to **C:\Edcognos\B5152\Course_Project**.
5. Create a new folder called **great_outdoors_warehouse_FINAL**, select it, and then click **OK**.
6. Click **Next**, click **Next** again, and then click **Publish**.
7. Click **Finish**, and then save and close the project.

Task 2. Examine the consolidated project.

1. Open the **great_outdoors_warehouse_FINAL** project.
2. Expand **Great Outdoors Warehouse>Data Source View>gosalesdw**.



Notice that the Employee Info and Marketing Info folders are no longer links. All metadata is now contained within this one project.

Also notice that the project maintains the original project name, **great_outdoors_warehouse_MASTER**. However the .cpf name is identified in the title bar, **great_outdoors_warehouse_FINAL**. To avoid confusion, it is recommended to rename the project to match the .cpf file.

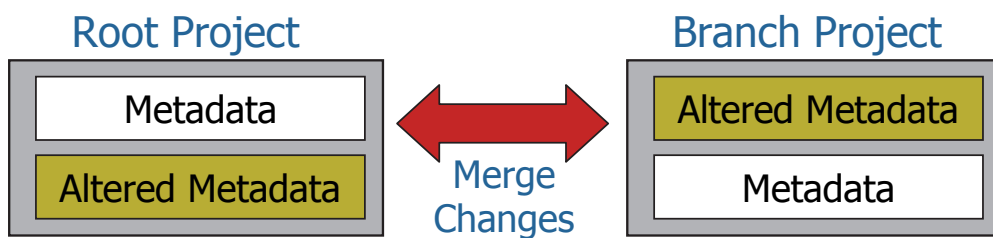
3. Rename the project to **great_outdoors_warehouse_FINAL**, and then save the project.

Results:

By publishing the entire great_outdoors_warehouse_MASTER project to the network, you consolidated all linked metadata into one project.

Branch a Model

- Enables parallel modeling
 - creates a copy of the project for simultaneous development on any portion of the model
- Unlimited branches
- Merge changes (bi-directional)



Unlike segmenting and linking, which allows for modelers to work on individual segments of a model independently, branching and merging allows multiple modelers to work on the same model in its entirety at the same time. They are not limited to just one segment and can make changes to any portion of the model. To do this, the project owner makes a copy of the root project, called a branch. A team member can modify the branch as required, independently of the root project. Branches can be merged back into the root project or from the root project into the branch as required.

Conflicts between the root project and a branch are resolved during the merge process.

Project branching is compatible with segmenting and linking.

Modelers should be in constant communication to ensure their modeling actions will not affect other branches or the root project.

Demo 4: Branch a Model, Make Changes and Merge Results

Purpose:

You would like to allow another modeler to work on your model at the same time as you. Use the project branching feature in Framework Manager to allow a second modeler to work on a branch of your project, and then merge the changes with your root project.

Component: Framework Manager

Project: great_outdoors_warehouse_FINAL

Task 1. Create a Branch of the root model.

This task is completed as modeler 1.

1. If it is not already open, open the **great_outdoors_warehouse_FINAL** project.
2. From the **Project** menu, click **Branch to**.
3. Change the branch project name to **great_outdoors_warehouse_FINAL Branch 1**, and then click **OK**.
4. Save and then close the **great_outdoors_warehouse_FINAL** project.

Task 2. Modify the branch model.

This task is completed as modeler 2.

1. Open the **great_outdoors_warehouse_FINAL Branch 1** model.
2. Expand **Great Outdoors Warehouse>Data Source View>gosalesdw>SLS_PRODUCT_DIM**.

3. Change the properties for the following items from **Fact** to **Attribute**:
 - **PRODUCT_TYPE_KEY**
 - **PRODUCT_NUMBER**
 - **BASE_PRODUCT_KEY**
 - **BASE_PRODUCT_NUMBER**
 - **PRODUCT_BRAND_KEY**
4. Right-click **SLS_RTL_DIM**, click **Launch Context Explorer** and then select **Show Related Objects**.
5. Double-click the relationship between **SLS_RTL_DIM** and **SLS_SALES_FACT**.
6. Change the cardinality for **SLS_SALES_FACT** to **0..n**.
7. Close the **Relationship Definition** dialog box and the **Context Explorer**.
8. Save and close the project.

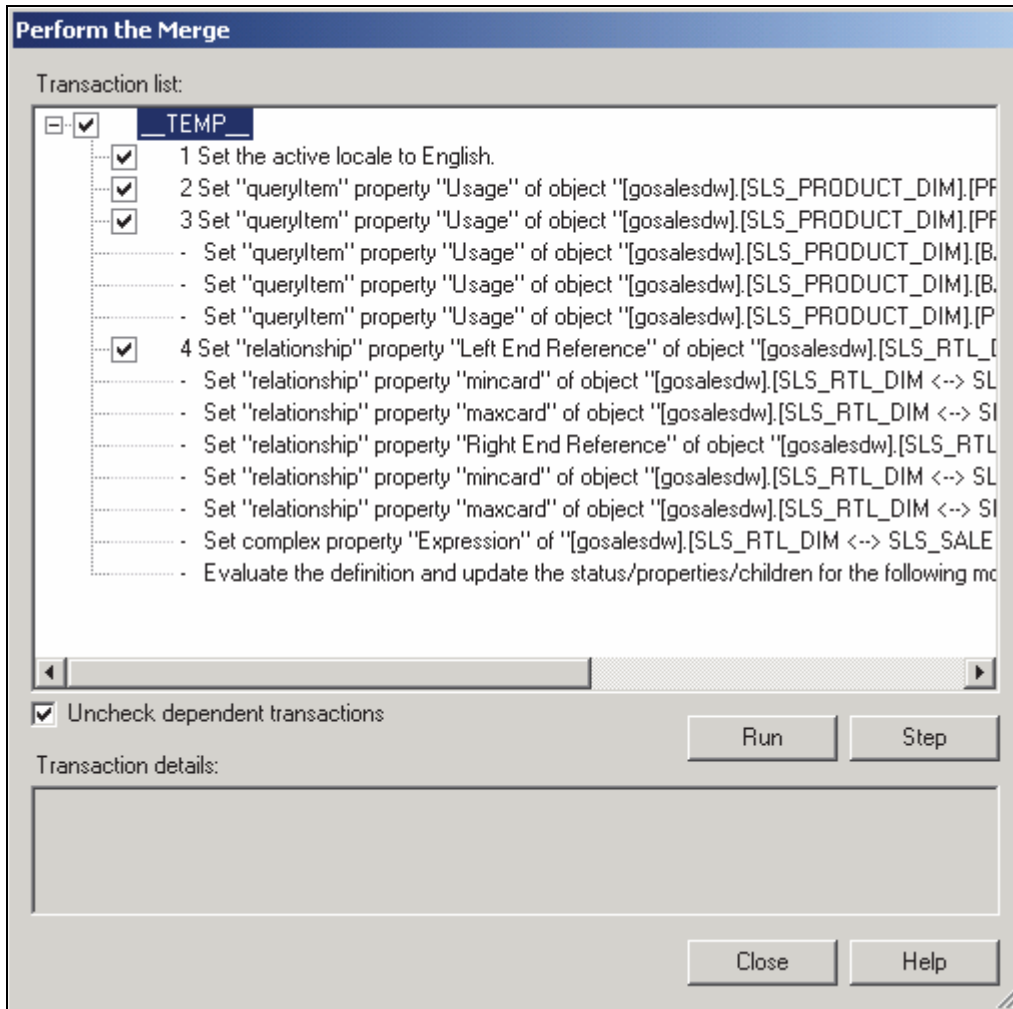
Task 3. Merge the branch into the root project.

This task is performed as modeler 1.

1. Open the **great_outdoors_warehouse_FINAL** model.
2. From the **Project** menu, select **Merge from**.

3. Navigate to the **great_outdoors_warehouse_FINAL Branch 1.cpf** and click **Open**.

The results appear as follows:



You are presented with a transaction list. You have the option to clear check boxes for transactions that you do not want updated in the root model. In your case you will accept all the changes. You can also choose to step through each transaction and then pause, or simply run all transactions.

4. Click **Run**.

All transactions are successfully merged into the root project. A backup project is created on the file system in the same location as the root project with a data timestamp. This allows you to revert back to the previous version if required.

After the branch project is merged, it will still exist on the file system. If it is no longer required, it is recommended that you delete the branch project after it is merged into the root project. New branches can be created as required.

5. Click **Accept**, view the same objects in the root project, and note the changes merged in from the branch project.
6. Save and close the project, and then close Framework Manager.

Results:

Using the project branching feature in Framework Manager, you were able to apply changes in a branched project and then merge those changes into the root project.

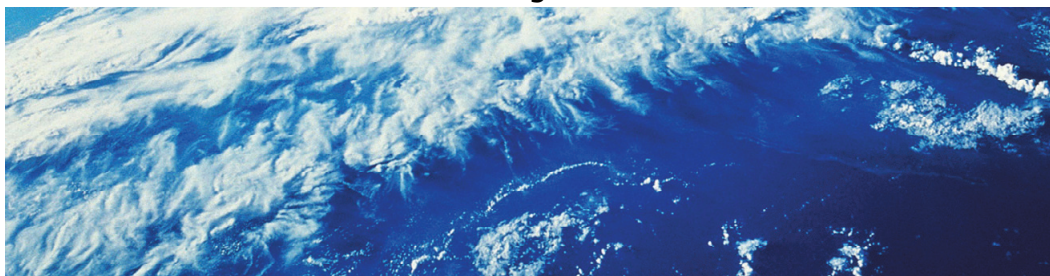
Summary

- You should now be able to:
 - implement repository control
 - segment and link a project
 - branch a project and merge results



Manage OLAP Data Sources

IBM Cognos BI



Business Analytics

© 2010 IBM Corporation

Objectives

- At the end of this module, you should be able to:
 - connect to an OLAP data source (cube) in a Framework Manager project
 - publish an OLAP model
 - publish a model with multiple OLAP data sources
 - publish a model with an OLAP data source and a relational data source

OLAP Data Sources in IBM Cognos

- IBM Cognos Business Intelligence provides full support for the analysis of PowerCubes and other OLAP data sources.

Revenue	2004	2005	2006	Years
Camping Equipment	\$20,471,328.88	\$31,373,606.46	\$37,869,055.58	\$89,713,990.92
Golf Equipment	\$5,597,980.86	\$9,598,268.88	\$10,709,215.84	\$25,905,465.58
Mountaineering Equipment	\$0.00	\$9,642,674.54	\$11,248,676.06	\$20,891,350.60
Outdoor Protection	\$1,536,456.24	\$988,230.64	\$646,428.04	\$3,171,114.92
Personal Accessories	\$7,144,797.52	\$10,955,708.04	\$13,793,960.30	\$31,894,465.86
Products	\$34,750,563.50	\$62,558,488.56	\$74,267,335.82	\$171,576,387.88

Other OLAP sources include:

- IBM Cognos Planning Analyst Models
- Microsoft Analysis Services
- IBM DB2 OLAP
- Hyperion Essbase

Connect to and Publish OLAP Data Sources

- Use Framework Manager or IBM Cognos Connection to connect to an OLAP data source.
- Cubes are published directly to the portal, without any modeling required.
- Use Framework Manager to:
 - create multi-cube packages
 - combine cubes and relational metadata in one package

You can use Framework Manager or IBM Cognos Connection to create data source connections to various cube data sources. When you import a cube into Framework Manager, no modeling is required since the modeling has already been done in an OLAP modeling tool such as IBM Cognos Transformer.

The package is then published directly to the portal, making it available to report authors.

Not only can you publish IBM Cognos PowerCubes from Framework Manager, you can also publish them directly in IBM Cognos Connection or from IBM Cognos Transformer.

Demo 1: Import and Publish an OLAP Data Source

Purpose:

Authors would like to be able to report from data stored in multiple OLAP sources. You will initially create a Framework Manager project and import a PowerCube.

Components: **Framework Manager, Analysis Studio**

Project: **OLAP_Model**

Package: **GOCube**

Task 1. Create a new project and import a cube.

1. In **Framework Manager**, close any projects that may be open, from the **File** menu, click **New**, and then name the new project **OLAP_Model**.
2. Set the **Location** to **C:\Edcognos\B5152\Course_Project\OLAP_Model**, and then click **OK**.
3. In the **Select Languages** dialog box, ensure that **English** is selected, and then click **OK**.
4. In the **Metadata Wizard** window, ensure that **Data Sources** is selected, and then click **Next**.
5. Click **New**, click **Next**, in the **Name** box, type **GOCube**, and then click **Next**.
6. Under **Type**, select **IBM Cognos PowerCube**, and then click **Next**.

The connectivity information you supply for a cube data source connection will depend on the type of cube source you are accessing. For example, PowerCubes require that you supply the filename and path for the cube.

To specify the location, you will use Windows Explorer to obtain the path and file name since the cube is local. For a network cube, use a UNC path.

7. Open **Windows Explorer** and navigate to **D:\Program Files\IBM\cognos\c10\webcontent\samples\datasources\cubes\PowerCubes\EN**.

Notice that there is a cube named great_outdoors_sales_en.mdc.

8. Copy the path from the **Address** box to the **Windows location** box in the **New Data Source Wizard**, and then type \.
9. Copy the name of the PowerCube (including file extension) to the **Windows location** box (after the \) in the **New Data Source Wizard**.
10. Scroll down, under **Testing**, click **Test the connection**, and then click **Test**.
A message appears indicating that the test was successful.
11. Click **Close**, and then click **Close** again.
12. Click **Finish**, and then click **Close**.
13. In the list of data sources, click **GOCube**, click **Next** and then click **Finish** twice.

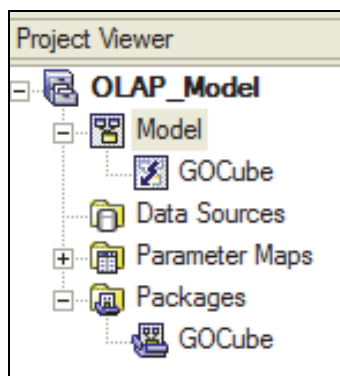
You are now prompted to create a package for this project. You will use the name GOCube for your package.

14. Click **No**, and then save the project.

Task 2. Examine objects and publish a package.

1. In the **Project Viewer** pane, expand **Model**, and then expand **Packages**.

The results appear as follows:



You can see that the GOCube model does not include any additional information. Everything needed to publish it is stored internally in the data source. You could have created your connection to the cube and your package entirely in IBM Cognos Connection, but you are doing it here so that you can later add additional cubes to the model.

2. Under **Packages**, right-click **GOCube**, and then click **Publish Packages**.
3. Clear the **Enable model versioning** check box, click **Next** twice, click **Publish**, and then, click **Finish**.
4. Save your project.

Task 3. Test the GOCube cube.

1. Launch **IBM Cognos Connection**, log in, and then launch **Analysis Studio** selecting the **GOCube** package.
2. Select **Default Analysis**, and then click **OK**.

The results appear as follows:

Rows:		Columns:	
Years		Products	
Revenue	Camping Equipment	Golf Equipment	Mountaineering Equipment
2004	332,986,338.06	153,553,850.98	
2005	402,757,573.17	168,006,427.07	107,099,659.94
2006	500,382,422.83	230,110,270.55	161,039,823.26
2007	352,910,329.97	174,740,819.29	141,520,649.70
Years	1,589,036,664.03	726,411,367.89	409,660,132.90

You have successfully published an IBM Cognos PowerCube, which can be used to analyze data and write reports in IBM Cognos.

3. Close **Analysis Studio** without saving the analysis.
4. Leave IBM Cognos Connection and Framework Manager open for the next demo.

Results:

You created a new Framework Manager project, connected to a PowerCube, and created a data source connection and package. You then published the package and tested it in Analysis Studio.

Demo 2: Import and Publish Multiple OLAP Data Sources

Purpose:

Authors would like to access financial data stored in a Microsoft Analysis Services cube. They would also like to be able to link data from this cube to the original Great Outdoors Company PowerCube. You will add this second cube to your model and then create a test report.

Components: Framework Manager, Analysis Studio

Project: OLAP_Model

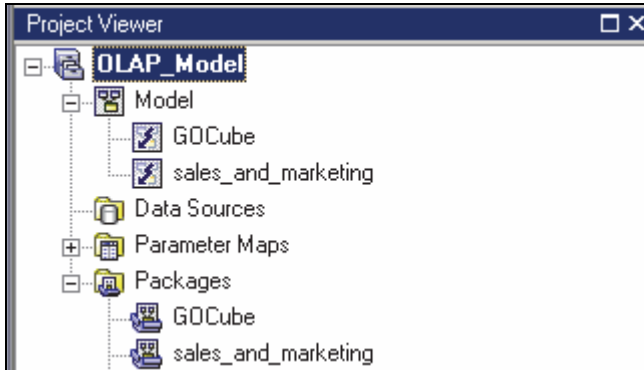
Package: GOCube

Task 1. Add the sales_and_marketing cube to the model.

1. In **Framework Manager**, in the **Project Viewer** pane, right-click **Model**, and then click **Run Metadata Wizard**.
2. In the **Metadata Wizard** window, ensure that **Data Sources** is selected, and then click **Next**.
3. In the **Metadata Wizard** window, click **sales_and_marketing**, and then click **Next**.
4. Click **Finish** to complete the wizard.
5. Click **Finish** to create the sales_and_marketing package.

- Click **No** when asked if you want to **Open the Publish Package Wizard**.

The results appear as follows:



Task 2. Add the sales_and_marketing package to the GOCube package and publish.


- In the **Project Viewer**, under **Packages**, double-click **GOCube**.
- Select **sales_and_marketing**, and then click **OK**.
- Publish the **GOCube** package.

Task 3. Create a report with the GOCube package.

- In **IBM Cognos Connection**, launch **Report Studio** selecting the **GOCube** package.
- Create a new **List** report.
- In the **Insertable Objects** pane, under **GOCube**, expand both **GOCube** and **sales_and_marketing**.

The dimensions for each cube appear. You will now create a report that displays data from both cubes.

- Expand **GOCube>Products>Products**, and then drag the **Product line** level to the report.
- Expand **Measures**, and then drag the **Revenue** to the report.

6. In the **Insertable Objects** pane, click the **Toolbox**  tab, and then drag a **List** object to the right of the list in the report, so that the list is added as a new column.


The results appear as follows:

Product line	Revenue	List				
<Product line>	<Revenue>					

7. On the **Source** tab, expand the **sales_and_marketing** cube.
8. Expand **Products>Products**, and then drag the **Product line** level to the report.
9. Expand **Measures**, and then drag **Gross Margin** and **Revenue** to the report.

The results appear as follows:

Product line	Revenue	List		
<Product line>	<Revenue>	Product line	Gross margin	Revenue
		<Product line>	<Gross margin>	<Revenue>
		<Product line>	<Gross margin>	<Revenue>
		<Product line>	<Gross margin>	<Revenue>

10. On the toolbar, click **Run Report** .

The results appear as follows:

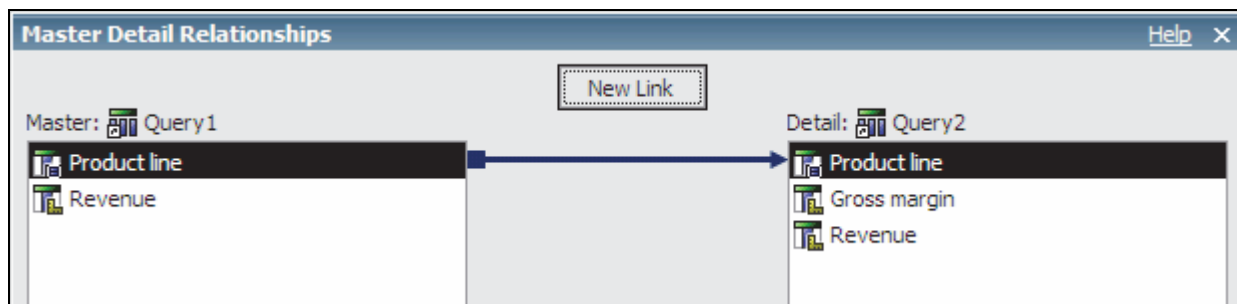
Product line	Revenue	List		
Camping Equipment	1,589,036,664.03	Product line	Gross margin	Revenue
		Camping Equipment	21.3%	1,589,036,664.03
		Golf Equipment	48.6%	726,411,367.89
		Outdoor Protection	93.1%	75,994,296.25
		Personal Accessories	14.9%	1,885,673,307.78
		Mountaineering Equipment	24.3%	409,660,132.90
Golf Equipment	726,411,367.89	Product line	Gross margin	Revenue
		Camping Equipment	21.3%	1,589,036,664.03
		Golf Equipment	48.6%	726,411,367.89
		Outdoor Protection	93.1%	75,994,296.25
		Personal Accessories	14.9%	1,885,673,307.78
		Mountaineering Equipment	24.3%	409,660,132.90

The report opens in Cognos Viewer. Gross Margin and Revenue is repeated for each Product line. You need to link the two list reports on Product line.

Task 4. Create a master-detail link.

1. Close Cognos Viewer.
2. In the inner list, click **Product line**, and then, from the **Data** menu, click **Master Detail Relationships**.
3. Click **New Link**.

The results appear as follows:



A link appears between the first column of each query (between Product line and Product line). This is the link that you want to create, so you do not have to modify it.

4. Click **OK**, and then run the report.

The results appear as follows:

Product line	Revenue	List		
Camping Equipment	1,589,036,664.03	Product line	Gross margin	Revenue
		Camping Equipment	21.3%	1,589,036,664.03
Golf Equipment	726,411,367.89	Product line	Gross margin	Revenue
		Golf Equipment	48.6%	726,411,367.89
Mountaineering Equipment	409,660,132.90	Product line	Gross margin	Revenue
		Mountaineering Equipment	24.3%	409,660,132.90

The margin and revenue data for each Product line appears in the proper rows. You have left the Product line column in the report for verification, but you would remove it from the report layout for the production report.

5. Close **Cognos Viewer**, and then close **Report Studio** without saving the report.
6. Close IBM Cognos Connection, and then close Framework Manager, without saving the changes.

Results:

You added an additional cube to your existing model and package. You then published the package and created a report that linked data from the new cube to the original cube.

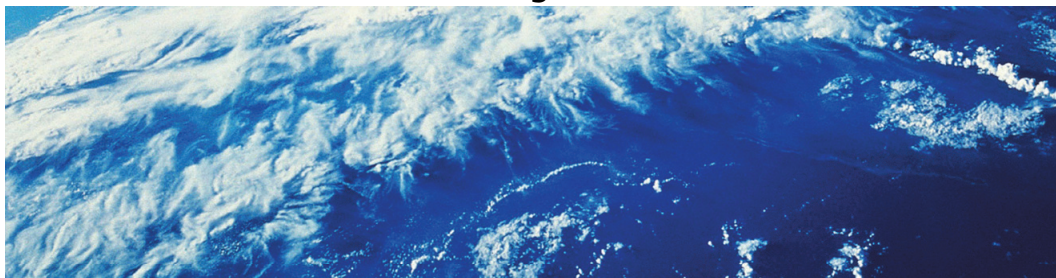
Summary

- At the end of this module, you should be able to:
 - connect to an OLAP data source (cube) in a Framework Manager project
 - publish an OLAP model
 - publish a model with multiple OLAP data sources
 - publish a model with an OLAP data source and a relational data source



Manage Framework Manager Packages

IBM Cognos BI



Business Analytics

© 2010 IBM Corporation

Objectives

- At the end of this module, you should be able to:
 - specify package languages and function sets
 - control model versioning
 - nest packages

What is a Framework Manager Package?

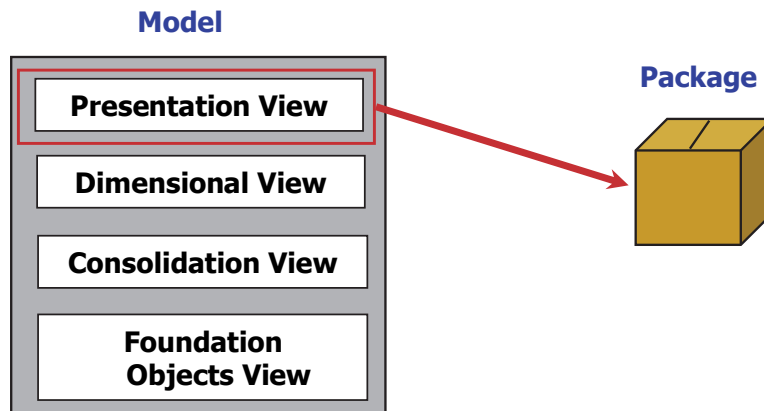
- A subset of the project metadata to be published to IBM Cognos Connection
- You can:
 - create different packages to meet different requirements
 - apply security to packages to restrict access
 - include other packages in a package (nesting)

Each package can contain a different set of folders, filters, query subjects, and query items. You can customize its contents to satisfy different reporting requirements and to set up a logical presentation of the metadata.

Nesting packages can save development and maintenance time.

Create and Modify Packages

- Include the model objects you want to publish in your package.



At any time, you may go back and edit the definition of your package by adding or removing objects. You must exercise caution when doing this as you may break reports based on a previous version of the model.

When you create a package, you can choose whether objects from the model will be included, excluded, or hidden, based on the requirements of reports authors. These three options are described as follows:

- Select - the object and its child objects can be used in reports by report authors
- Hide - the object and its child objects cannot be used by report authors, but objects available to report authors that reference the hidden objects can. With this option, you will not receive informational messages when publishing your package stating that underlying objects will be published because other objects in the package reference them.
- Unselect - the object and its child objects are not published. It cannot be used for reports and cannot be selected by report authors. If other objects in the package reference the unselected object, the object will be included and hidden in the package and you will receive informational messages stating this.

Languages and Function Sets

- Specify which languages are to be included in a package.
 - Each language added will add additional metadata that has been translated.
- Specify which function sets to include in a package.
 - Add vendor specific function sets to match the data sources used in a package.

In the case where you have modeled for a multilingual audience, you can specify the languages to be published with a package. You must add languages to the project before you can add them to a package, and must translate the metadata for the model objects.

If you have multiple data sources in your model that are heterogeneous, publish your package with the appropriate function sets so that report authors can leverage them while authoring reports.

Publish Packages

- When you publish a package, you can choose to:
 - publish to the IBM Cognos content store or to a network location
 - externalize query subjects
 - verify the package

When you publish a package to the IBM Cognos server, you make it available to report authors with the appropriate security rights.

Publishing to the network lets you back up or share all or a portion of your model with other metadata modelers.

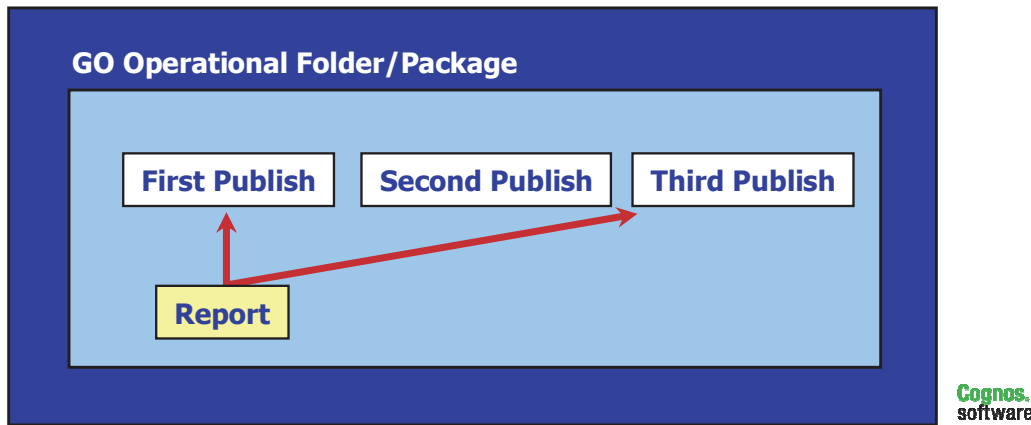
You can choose to externalize query subjects to make metadata and data available from the underlying data source available for use in other applications.

To avoid potential problems, select the verify package option in the Publish Wizard to check for errors.

Set Model Version Control

- Select the number of versions of the model you want to retain on the IBM Cognos server.

IBM Cognos Connection



Model versioning allows for multiple versions of the same model to be maintained in IBM Cognos. This allows for changes to the model without immediately affecting existing reports. Existing reports will still use the original version of the model allowing authors time to repair any damaged reports before the original model expires.

In the slide example, if the number of model versions to retain is set to 2, and you create a report based on the first publish of the package, that report will continue to use the first version of the model, even after the second publish. On the third publish however, the report will automatically use the latest version. Any reports created or modified by report authors will use the most recent version of the model.

Demo 1: Control Model Versions

Purpose:

The Sales and Marketing (conformed) package is only required in a few regions. Therefore, you will reduce the languages published with the project to those required for the regions. You will also reduce the function sets published with the package to just the one function set required for the database vendor they are using.

You will also enable model versioning so that authored reports can continue to run even though the model has changed. This will allow authors some time to fix their reports. You will enable model versioning and test the results.

Components: Framework Manager, Query Studio, Cognos Viewer

Project: great_outdoors_warehouse


Package: Sales and Marketing (conformed)

Task 1. Specify Package Languages and Function List.

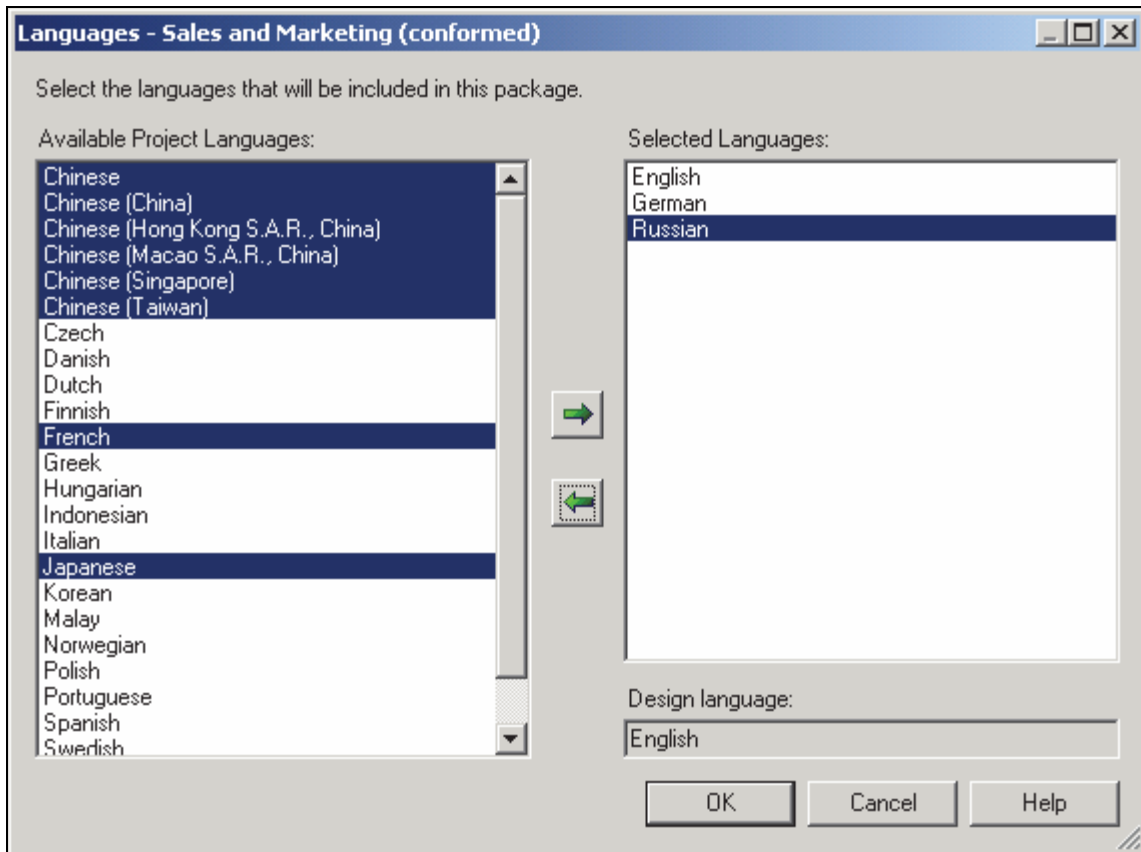
1. In **Framework Manager**, close any projects that may be open, and then open the **great_outdoors_warehouse** project located at **C:\Edcognos\B5152\CBIFM-Start Files\Module 24\great_outdoors_warehouse**.
2. In the **Project Viewer** pane, expand the **Packages** folder, and then click **Sales and Marketing (conformed)**.

This package is used primarily in Germany, Canada and Russia for drill through from dimensional data sources. You will limit the languages published with this package for the appropriate regions.

3. From the **Actions** menu, point to **Package**, and then click **Specify Package Languages**.

4. In the **Selected Languages** pane, remove all languages except **English**, **German**, and **Russian** using the left pointing green arrow  button.

The results appear as follows:



Note: By default, all languages available in a project will be added to a package when you create it. If languages are added to the project after you create the package, and you want those languages included in your package, you must add them to the package.

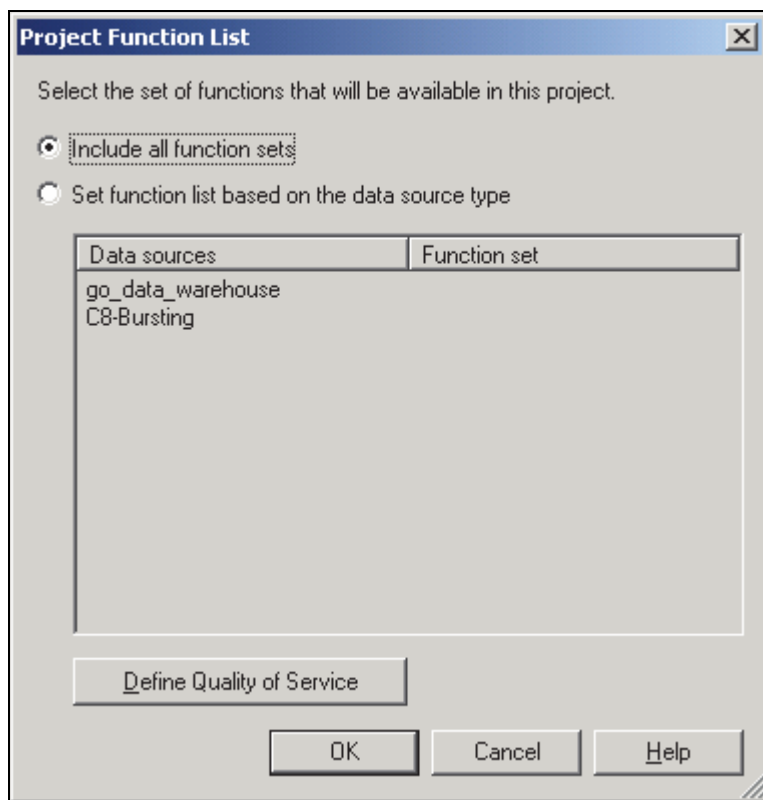
You can also specify package languages for several packages at a time. To do this, in the model, click the Packages folder, from the Actions menu, point to Package, and then click Specify Package Languages. Also, from the Packages sub menu, you can also click Explore Packages to view the contents of each package and any object security specified.

5. Click **OK**.

The database vendor for these regions will be using is DB2. You will limit this package to the DB2 function set. First you will examine where you can define which function set is associated with a data source.

6. From the **Project** menu, click **Project Function List**.

The results appear as follows:



The Project Function List dialog box appears. Here you can decide which functions sets will appear by default when creating a new package. You can choose to include all function sets, or define a function set for each data source in the project. If you choose the second option, by default, only defined function sets will be selected for your package based on the data sources referenced in the package.

7. Click **Set function list based on the data source type**, and then click in the **Function set** column of the **go_data_warehouse** row.

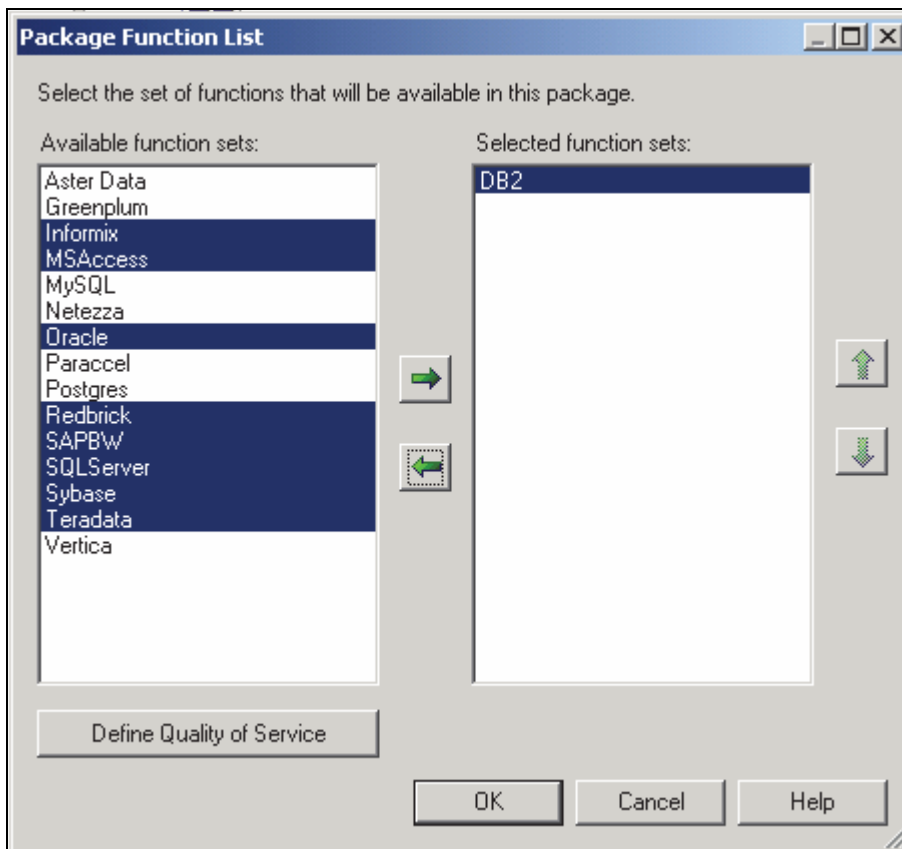
A list of database-specific function sets appears.

8. Click **DB2**.
9. Click **OK**.

When this package was initially created, the setting was to include all function sets. You will configure this package appropriate to your data source.

10. From the **Actions** menu, point to **Package**, and then click **Specify Package Function List**.
11. Remove all function sets except **DB2**.

The results appear as follows:



12. Click **OK**.

Now the next time the Sales and Marketing (conformed) package is published, it will be reduced in size since it will contain fewer multilingual metadata and fewer function sets. It will only contain what is required by the authors.

Task 2. Enable model versioning.

To ensure that authors have time to fix their reports when changes are made to the model, you will enable model versioning.

1. Right-click the **Sales and Marketing (conformed)** package, and then click **Publish Packages**.
2. Under **Select publish location**, select the **Enable model versioning** check box, and then in the **Number of model versions to retain** box, type 2.
3. Select the **Delete all previous model versions** check box, click **Next**, click **Next** again, and then clear the **Verify the package before publishing** check box.

In this case, selecting Delete all previous versions will ensure that you are working with a fresh version of the package for demonstration purposes. This option can be used to force reports and analyses to use the latest version of the model.

4. Click **Publish**, and then click **Finish**.


Task 3. Create a report based on the Sales and Marketing (conformed) package.

1. Launch **IBM Cognos Connection**, log on, launch **Query Studio** selecting the **Sales and Marketing (conformed)** package.
2. In the **Insert Data** menu, expand **Sales and Marketing (conformed)>Products**, and then drag **Product** to the report.

- From **Measures>Sales fact**, add **Revenue** and **Planned revenue** to the report.

The results appear as follows:

Product	Revenue	Planned revenue
TrailChef Water Bag	23,057,141.46	28,395,176.52
TrailChef Canteen	11,333,518.65	12,561,922.23
TrailChef Kitchen Kit	19,535,825.83	20,626,722.2
TrailChef Cup	5,702,502.7	6,632,370.18
TrailChef Cook Set	41,184,274.9	44,700,935.4
TrailChef Deluxe Cook Set	53,195,154.45	57,353,881.92
TrailChef Single Flame	43,189,819.56	45,837,137.61

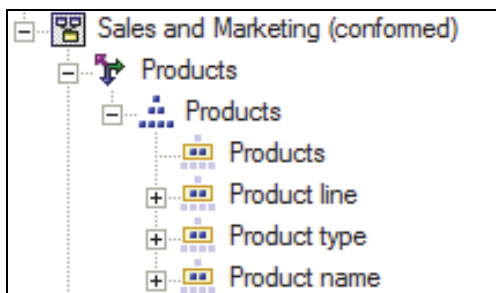
- On the toolbar, click **Save** , in the **Name** box, type **Model Versioning Test**, and then click **Save**.
- Return to **IBM Cognos Connection**.

Task 4. Modify the project, and then re-publish the Sales and Marketing (conformed) package.

Authors have requested that the Product level in the Products regular dimension be renamed to Product name to be consistent with the naming convention of the parent levels.

- In **Framework Manager**, in the **Project Viewer** pane, expand **go_data_warehouse>Sales and Marketing (conformed)>Products>Products**.
- Rename the **Product** level to **Product name**.

The results appear as follows:




3. Save the project, and then, publish the **Sales and Marketing (conformed)** package. Use the same options as previously chosen, except leave the **Delete previous model versions** check box deselected.
4. Click **Next**, click **Next** again, and then click **Publish**.

A message box appears stating that a previous version of the model already exists, and prompting you to add an additional version.

5. Click **Yes**, and then click **Finish**.

You now have two versions of the Sales and Marketing (conformed) package on the IBM Cognos server.

Task 5. Run the saved report.

1. On the **Public Folders** page of **IBM Cognos Connection**, click the **Sales and Marketing (conformed)** folder, and then click **Run with options - Model Versioning Test** .

2. Click **Run**.

The report runs and opens in Cognos Viewer without issue. This report has run against the original version of the package, which is different from the package you just published.

3. Return to **IBM Cognos Connection**.

Task 6. Re-publish the Sales and Marketing (conformed) package and run the saved report.

1. In **Framework Manger**, re-publish the **Sales and Marketing (conformed)** package using the same options in the Publish Wizard.

A message box appears stating that two model versions of the model already exist, that the maximum allowable is two, and that publishing will overwrite one of the versions. The message also prompts you to continue with the publish.

2. Click **Yes**, and then click **Finish**.

3. In **IBM Cognos Connection**, re-run the report as in the previous task.

An error message appears.

4. Click **Details**.

The message indicates that

QE-DEF-0359 The query contains a reference to at least one object '[Sales and Marketing (conformed)].[Products].[Products].[Product]' that does not exist.

5. Click **OK**, and then click **Cancel**.

Task 7. Open the saved report in Query Studio.

1. Click the **Model Versioning Test** report to open it in **Query Studio**.

A message appears indicating that a more recent version of the package exists and that if you want to complete the update, you must save the report.

2. Click **OK**.

The results appear as follows:

Revenue	Planned revenue
4,686,775,768.85	4,919,342,361.94

The Product level column is no longer valid and does not appear in the report. You will need to re-add this column to the report to repair it.

- Under **Menu**, click **Insert Data**, expand **Sales and Marketing (conformed)**>**Products**, and then drag **Product name** to the report as the first column.

The results appear as follows:

Product name	Revenue	Planned revenue
<u>TrailChef Water Bag</u>	23,057,141.46	28,395,176.52
<u>TrailChef Canteen</u>	11,333,518.65	12,561,922.23
<u>TrailChef Kitchen Kit</u>	19,535,825.83	20,626,722.2
<u>TrailChef Cup</u>	5,702,502.7	6,632,370.18
<u>TrailChef Cook Set</u>	41,184,274.9	44,700,935.4
<u>TrailChef Deluxe Cook Set</u>	53,195,154.45	57,353,881.92
<u>TrailChef Single Flame</u>	43,189,819.56	45,837,137.61

- On the toolbar, click **Save**, and then return to **IBM Cognos Connection**.

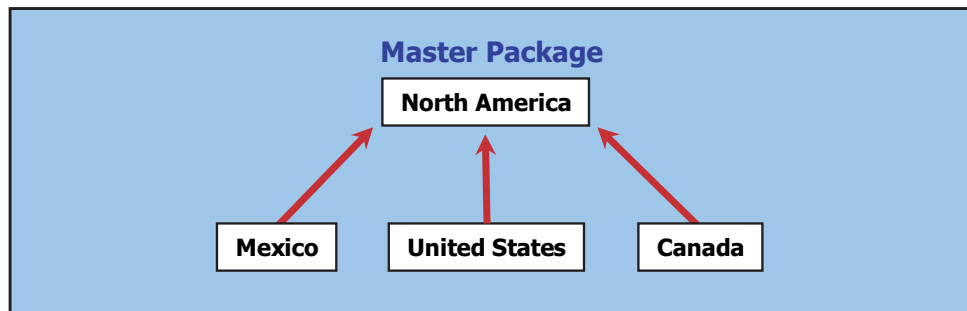
Results:

You reduced the number of languages and function sets published with the **Sales and Marketing (conformed)** package, and enabled and tested model versioning to ensure reports can continue to run even though the model may change.

Nest Packages

- When you create nested packages, you create a master package that is based on other packages.

Framework Manager Model Packages

**Cognos.**
software

© 2010 IBM Corporation

Use nested packages to reuse model information.

Nested packages save time, are easier to maintain, and let you publish only the master package to make all referenced packages available to report authors.

Demo 2: Nest Packages

Purpose:

One of the quality assurance teams that tests your packages wants to combine the GO Data Warehouse (query) and GO Data Warehouse (analysis) packages into one package for testing and comparison purposes. You will use a nested package to fulfill this request.

Components: Framework Manager, Business Insight Advanced

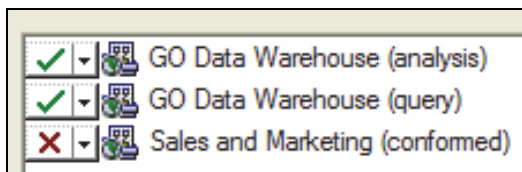
Project: great_outdoors_warehouse

Package: GO Data Warehouse (query and analysis)

Task 1. Create a nested package.

1. In **Framework Manager**, create a new package called **GO Data Warehouse (query and analysis)**, and then click **Next**.
2. In the **Define Objects** dialog box, select **Using existing packages**.
3. Click the **GO Data Warehouse (analysis)** and the **GO Data Warehouse (query)** packages

The results appear as follows:



4. Click **Next**, and then click **Finish**.
5. Click **Yes**, deselect **Enable model versioning**, and then click **Next** twice
6. Clear the **Verify the package before publishing** check box, click **Publish**, and then click **Finish**.
7. Save the project.

Task 2. View the package in Business Insight Advanced.

1. In **IBM Cognos Connection**, ensure you are in **Public Folders**, and then launch **Business Insight Advanced** selecting the **GO Data Warehouse (query and analysis)** package for a **List** report.

The results appear as follows:



Quality assurance staff now has access to both the query and analysis views of the metadata, and can quickly perform tests on either view without switching packages.

2. Close **IBM Cognos Connection**, do not save changes, and then, in **Framework Manager**, close the **great_outdoors_warehouse** project.

Results:

By nesting existing packages, you created a package that re-used two different presentation views of the metadata in the project for use in comparative testing by the quality assurance team.

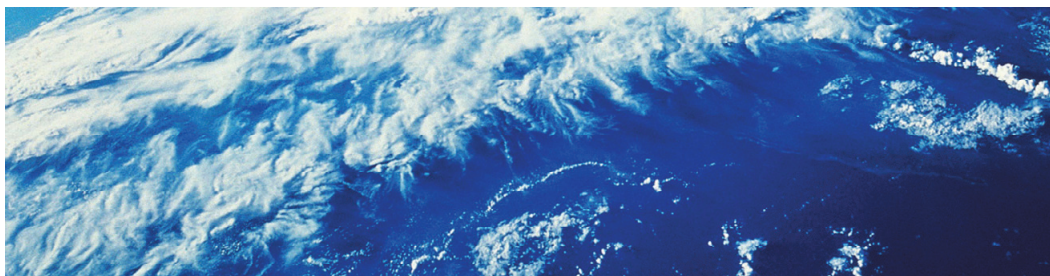
Summary

- You should now be able to:
 - specify package languages and function sets
 - control model versioning
 - nest packages



Employ Additional Framework Manager Modeling Techniques

IBM Cognos BI



Business Analytics

© 2010 IBM Corporation

Objectives

- At the end of this module, you should be able to:
 - leverage a user defined function
 - set the order of operations in a model calculation
 - externalize query subjects
 - prepare IBM Cognos content for use as a data source in Transformer
 - create query sets

Leveraging User-Defined Functions

- User-defined functions:
 - return a single value per request (per row)
 - can be imported or referenced in the SQL of the query subject definition

Planned Revenue Calculation

Calc_Planned_Revenue([gosales].[ORDER_DETAILS].[QUANTITY],
[gosales].[ORDER_DETAILS].[UNIT_PRICE])



User Defined Function Reference

When you reference a user-defined function in a query subject, it will be represented as a query item and will return a value for each row of data as calculated by the function.

If you wished to reference the user-defined function in the slide example in the SQL of the query subject rather than importing it into the project, the syntax would appear as shown below:

select

ORDER_DETAILS.ORDER_DETAIL_CODE,

.....

dbo.Calc_Planned_Revenue(ORDER_DETAILS.QUANTITY,ORDER_D
ETAILS.UNIT_PRICE) as "Planned Revenue"

from [GOSALES].ORDER_DETAILS

Examine the Regular Aggregate Property

- The Regular Aggregate property has the following settings:
 - automatic
 - average
 - calculated (applies only to standalone calculations)
 - count
 - maximum
 - minimum
 - sum

The above are options for relational data source query items with their Usage property set to Fact.

The setting made in the model will become the default behavior in the reporting studios.

Set Order of Operations for Model Calculations

- If you have a model calculation ($A*B$) set to calculated, where A and B are set to sum:
 - the behavior in Query Studio and Report Studio will be to summarize first and then calculate
 - $\text{sum}(A) * \text{sum}(B)$

Summary
Row

A	B	A * B
5	10	50
10	5	50
15	15	100

Regular Aggregate
Property Set
to Sum

A * B
50
50
225

Regular Aggregate
Property Set
to Calculated
Cognos.
software

© 2010 IBM Corporation

The Calculated setting in the Regular Aggregate property controls the order of operations in standalone model calculations.

Demo 1: Set Order of Operations for a Model Calculation

Purpose:

Authors would like to see margin values in their reports. Currently a margin query item calculation is available to them, but it is performing the margin calculation first and then aggregating the values, which is returning undesired results. They would like to see the underlying values aggregated first and then perform the margin calculation.

To accomplish this, you will create a stand-alone margin calculation, since this type of operation is only supported for stand-alone calculations, and set the Regular Aggregate property accordingly.

Components: Framework Manager, Query Studio

Project: GO Operational

Package: GO Operational (query)

Task 1. Test existing Margin query item.

1. Launch **Framework Manager**, and then open the **GO Operational** project located at **C:\Edcognos\B5152\CBIFM-Start Files\Appendix A\GO Operational**.
2. Expand **Foundation Objects View**, expand **gosales** namespace, and then expand **Sales Fact**.

3. Select and test the following items:

- **Revenue**
- **Gross Profit**
- **Margin**

The results appear as follows:

Test results			
	Revenue	Gross Profit	Margin
	8624.64	4625.92	0.53636093796379
	9411.6	4840.12	0.51427174975562
	18032.22	5072.22	0.28128649717007
	6690.8	2643.64	0.39511568123393

These are un-aggregated values. The Margin calculation is Gross Profit divided by Revenue. The margin values are correct. You will now test the items with Auto Sum enabled.

4. Select **Auto Sum**, and then click **Test Sample**.

The results appear as follows:

Test results			
	Revenue	Gross Profit	Margin
	4686775768.85	1924834994.68	194030.904167937

Notice the margin value is not what is expected. All the detail records are being calculated first and then aggregated. The value shown in the Margin column is the sum of all the margin values seen in the un-aggregated result set in the previous test. The expected result would be the Gross Profit value above divided by the Revenue value above for a margin of 0.41.

5. Click **Close**.

Task 2. Create a stand-alone Margin calculation and set its Regular Aggregate property.

1. Double-click **Sales Fact**, right-click **Margin**, and then click **Convert to Stand-alone Calculation**.

The Margin query item will no longer be required in the Sales Fact query subject. Setting the order of operations to aggregate first and then calculate is only valid for stand-alone calculations. The calculation cannot be restricted by the scope of a query subject container.

2. Delete the **Margin** query item.

A message appears indicating that the Margin query item in the Sales Fact query subject in the Consolidation View will be invalidated. You will also delete that query item in the next few steps.

3. Click **OK**, and then click **OK** again.

The stand-alone Margin calculation appears at the bottom of the gosales namespace.

4. In the **Consolidation View**, expand **Sales Fact** and delete the **Margin** query item.

A message appears saying that the Margin measure in the Sales Fact regular dimension in the Dimensional View namespace will be invalidated. You will delete this object in the next few steps.

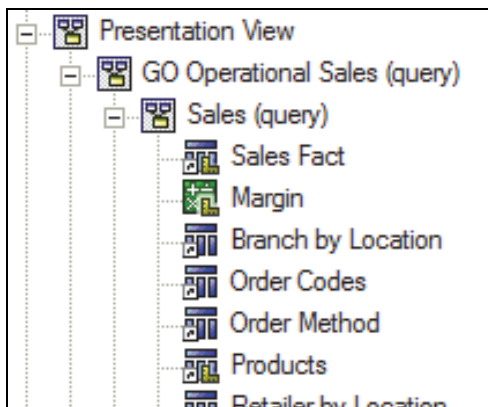
5. Click **OK**.

6. Expand **Dimensional View>Sales Fact**, and then delete the **Margin** query item.

Note: If you want to make the Margin calculation available for your analysis package, simply make the stand-alone calculation available in that package.

Task 3. Change the Regular Aggregate property for the Margin calculation and test in Query Studio.

1. In the **Project Viewer**, select the stand-alone **Margin** calculation, and then in the **Properties** pane, change the **Regular Aggregate** property to **Calculated**.
2. Move the stand-alone **Margin** calculation to the **Sales (query)** namespace in the **GO Operational Sales (query)** namespace, in the **Presentation View**.
3. Arrange the stand-alone **Margin** calculation as shown below:



4. Publish the **GO Operation (query)** package.
5. Launch **IBM Cognos Connection**, log on, and then launch **Query Studio** selecting the **GO Operational (query)** package.
6. In the **Insert Data** menu, expand **Sales (query)>Sales Fact**, and then select the following items:
 - **Revenue**
 - **Gross Profit**
 - **Margin**

7. Drag the selected items onto the report.

The results appear as follows:

Revenue	Gross Profit	Margin
\$4,686,775,768.85	\$1,924,834,994.68	41%

Notice the aggregated Margin value is now correct. The detail values for Gross Profit and Revenue are aggregated first and then calculated. This value has been formatted as a percentage in Framework Manager. If it had not been formatted, it would have appeared as 0.41069491898314.

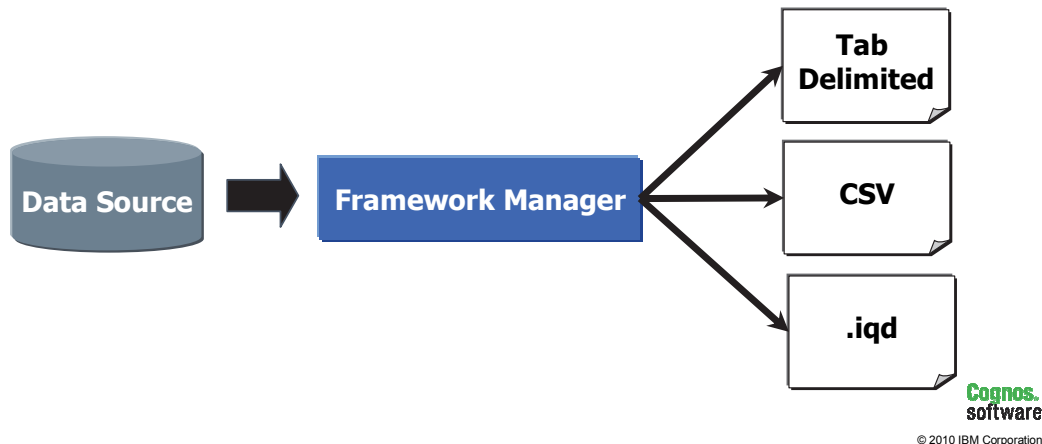
8. Close your browser without saving the report.
9. In **Framework Manger**, save and close the project.

Results:

You created a stand-alone margin calculation and set the Regular Aggregate property to Calculated to perform an aggregation before your calculation.

Externalize Query Subjects

- Framework Manager can export model metadata/data in various formats for use in other applications.



When publishing a package, you can externalize query subjects so that you can leverage them as data sources in other applications. For IBM Cognos Business Intelligence Transformer, we recommend that you access this content through IBM Cognos reports or packages.

To externalize a query subject, you must perform the following tasks:

1. Set the Externalize Method property for the query subject.
2. Use the Publish Package wizard to generate the required files.

Demo 2: Externalize Query Subjects to CSV Format

Component: Framework Manager

Project: great_outdoors_sales

Purpose:

A request has come in to make data produced by some of your query subjects available for use in other applications. To fulfill this request, you will externalize query subjects from your model to tab delimited files. They would like data for time, products, and sales targets. The sales targets data should be summarized at the month level and the product data should only go down to the product type level.

Task 1. Create model query subjects and set properties to externalize query subjects.

1. In Framework Manager, open the **great_outdoors_sales** project located at **C:\Edcognos\B5152\CBIFM-Start Files\Appendix_A\great_outdoors_sales**.

There is already one query subject that can be leveraged.

2. Expand **go_sales>Business view**.
3. Select **Time dimension**, and then in the **Properties** pane, change the **Externalize Method** property to **tab**.

Naming conventions have been applied to this object and some other modeling techniques. This is a good candidate to provide time data when you externalize your query subjects.

You will now create model query subjects to fulfill the requirements of the request made. Sales targets are at the product line level not the product level. So you will create a **Product Type (externalize)** model query subject that only contains query items down to the product line level. You will also create a **Sales Target (externalize)** model query subject that only contains the month key, product type key, and sales target values so that you can aggregate the values at the month level and product type level.

4. Expand **Database view>gosales**.
5. In the **Model query subjects (gosales)** folder, create a new model query subject called **Product Type (externalize)**.
6. In the **Available Model Objects** pane, expand **Database view>gosales>Model query subjects (gosales)>Product**, and then drag the following items to the **Query Items and Calculations** pane:
 - **Product line code**
 - **Product line**
 - **Product type code**
 - **Product type**
7. Click **OK**.
8. In the **Model query subjects (gosales)** folder, create a new model query subject called **Sales Target (externalize)**.
9. In the **Available Model Objects** pane, expand **Database view>gosales>Model query subjects (gosales)>Sales target**, and then drag the following items to the **Query Items and Calculations** pane:
 - **Month key**
 - **Product type code**
 - **Sales target**

10. Click **OK**.
11. In the **Project Viewer**, select the **Sales Target (externalize)** and **Product Type (externalize)** model query subjects.
12. If necessary, in the **Properties** pane, scroll to the right until you see the **Externalize Method** property and then set it to **tab** for both query subjects.

To ensure the sales target values are aggregated and you get only unique product type values, you must set the **Externalize Auto Summary** property for the **Sales Target (externalize)** and **Product Type (externalize)** model query subjects. This is true for product types since the underlying model query subject goes down to the product level. This causes repeating values at the product type level.

13. In the **Properties** pane, change the property setting for **Externalize Auto Summary** for both objects to **true**.

Task 2. Create a package and externalize the model query subjects.

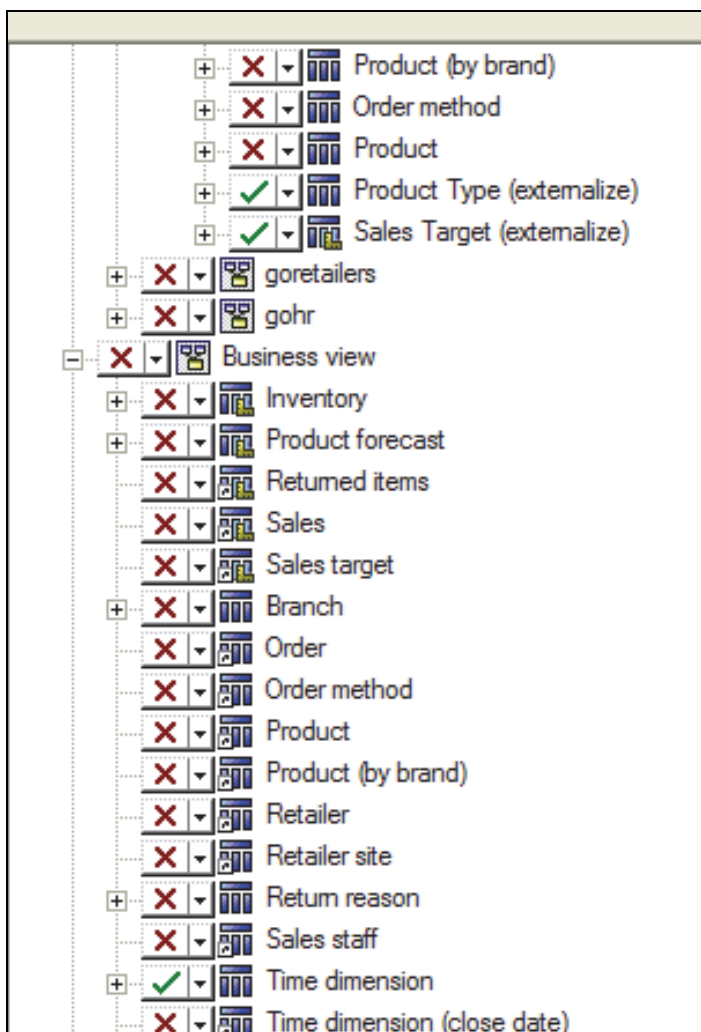
1. Right-click the **Packages** folder, point to **Create**, and then click **Package**.
2. In the **Name** box, type **Externalize Query Subjects**, and then click **Next**.
3. Select the check mark beside the **go_sales** namespace.

All items are now deselected.

4. Click the **X** beside the model query subjects you set to externalize in the previous task.

- Product Type (externalize)
- Sales Target (externalize)
- Business view > Time dimension

The results appear as follows:



These items are now selected and will be the only items included in the package.

5. Click **Finish**, and then click **Yes** to opening the **Publish Package** wizard.
When publishing, you have to choose to publish to the IBM Cognos content store (IBM Cognos Connection) or to a Network location. You will choose Network location in this demo, because you do not want this package to be available in IBM Cognos Connection.
6. Deselect **Enable model versioning**, click **Location on the network**, and then beside the **Network location** box, click **Browse**.
7. Navigate to **C:\Edcognos\B5152\Course_Project**.
8. Create a new folder called **Externalized Query Subjects Model**, select it, and then click **OK**.
9. Click **Next**, and then click **Next** again.
10. Select **Generate the files for externalized query subjects**, click **Browse** and then navigate to **C:\Edcognos\B5152\Course_Project** and create a folder called **Modeling Techniques**.
11. Click **OK**, clear the **Verify the model before publishing** check box, and then click **Publish**.

The results appear as follows:

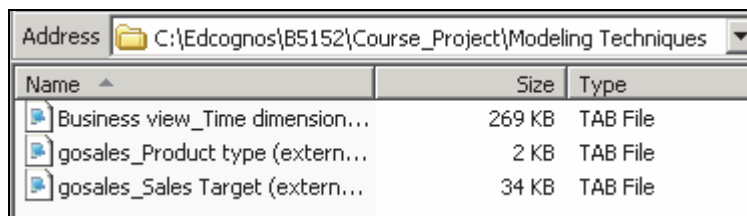
Query Subject ID	Method	Path
[gosales].[Sales Target (externalize)]	tab	C:\Edcognos\B5152\Course_Project\Modeling Techniques\gosales_Sa
[Business view].[Time dimension]	tab	C:\Edcognos\B5152\Course_Project\Modeling Techniques\Business vi
[gosales].[Product type (externalize)]	tab	C:\Edcognos\B5152\Course_Project\Modeling Techniques\gosales_Pri

12. Click **Finish**.

Task 3. Examine the generated files.

1. Open **Windows Explorer**, and then navigate to **C:\Edcognos\B5152\Course_Project\Modeling Techniques**.

The right pane of Windows Explorer appears as shown below:



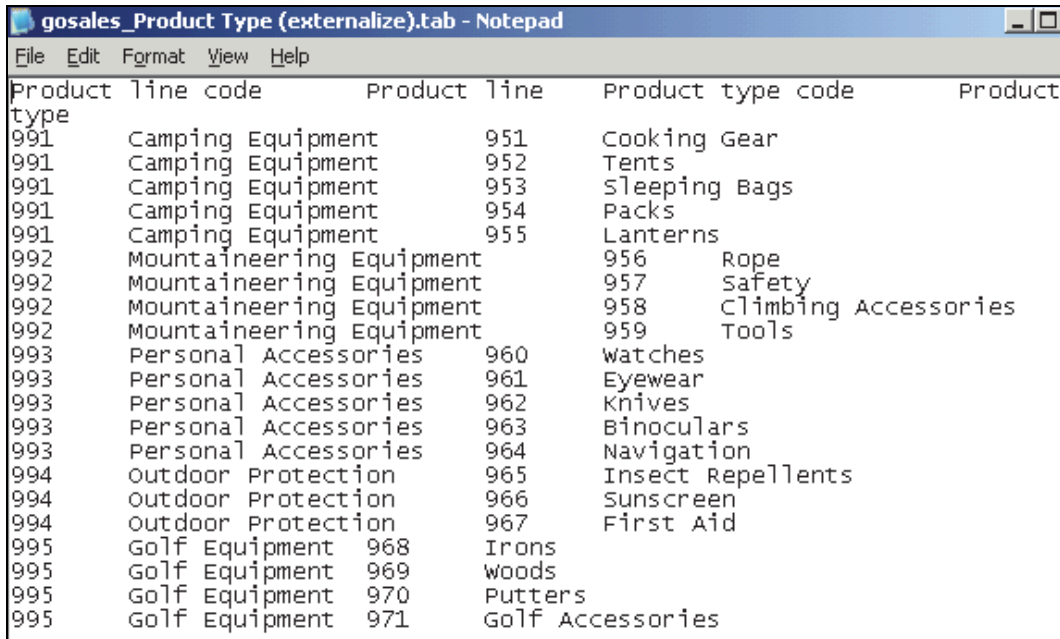
Name	Size	Type
Business view_Time dimension...	269 KB	TAB File
gosales_Product type (extern...	2 KB	TAB File
gosales_Sales Target (extern...	34 KB	TAB File

The publishing operation, based on the choices we made in the Publish Package wizard, has generated the tab files and the Framework Manager project files, located under Course_Project. The name of the tab files are based on the name of the namespace that contains the externalized query subject and the name of the externalized query subject.

2. Double-click **gosales_Product Type (externalize).tab**.

3. Select **Select the program from a list**, select **Notepad**, and then click **OK**.

The results appear as follows:



Product line code	Product line	Product type code	Product type
991	Camping Equipment	951	Cooking Gear
991	Camping Equipment	952	Tents
991	Camping Equipment	953	Sleeping Bags
991	Camping Equipment	954	Packs
991	Camping Equipment	955	Lanterns
992	Mountaineering Equipment	956	Rope
992	Mountaineering Equipment	957	Safety
992	Mountaineering Equipment	958	Climbing Accessories
992	Mountaineering Equipment	959	Tools
993	Personal Accessories	960	Watches
993	Personal Accessories	961	Eyewear
993	Personal Accessories	962	Knives
993	Personal Accessories	963	Binoculars
993	Personal Accessories	964	Navigation
994	Outdoor Protection	965	Insect Repellents
994	Outdoor Protection	966	Sunscreen
994	Outdoor Protection	967	First Aid
995	Golf Equipment	968	Irons
995	Golf Equipment	969	Woods
995	Golf Equipment	970	Putters
995	Golf Equipment	971	Golf Accessories

The first row contains tab delimited column headings and the remaining rows contain tab delimited data. Only unique values are returned for product types since the data set was aggregated.

4. Close **Notepad**, and then open the **gosales_Sales Target (externalize).tab** file to view it.

Again, the first row contains tab delimited column headings and the remaining rows contain tab delimited data. The sales target values are aggregated to the month and product type levels.

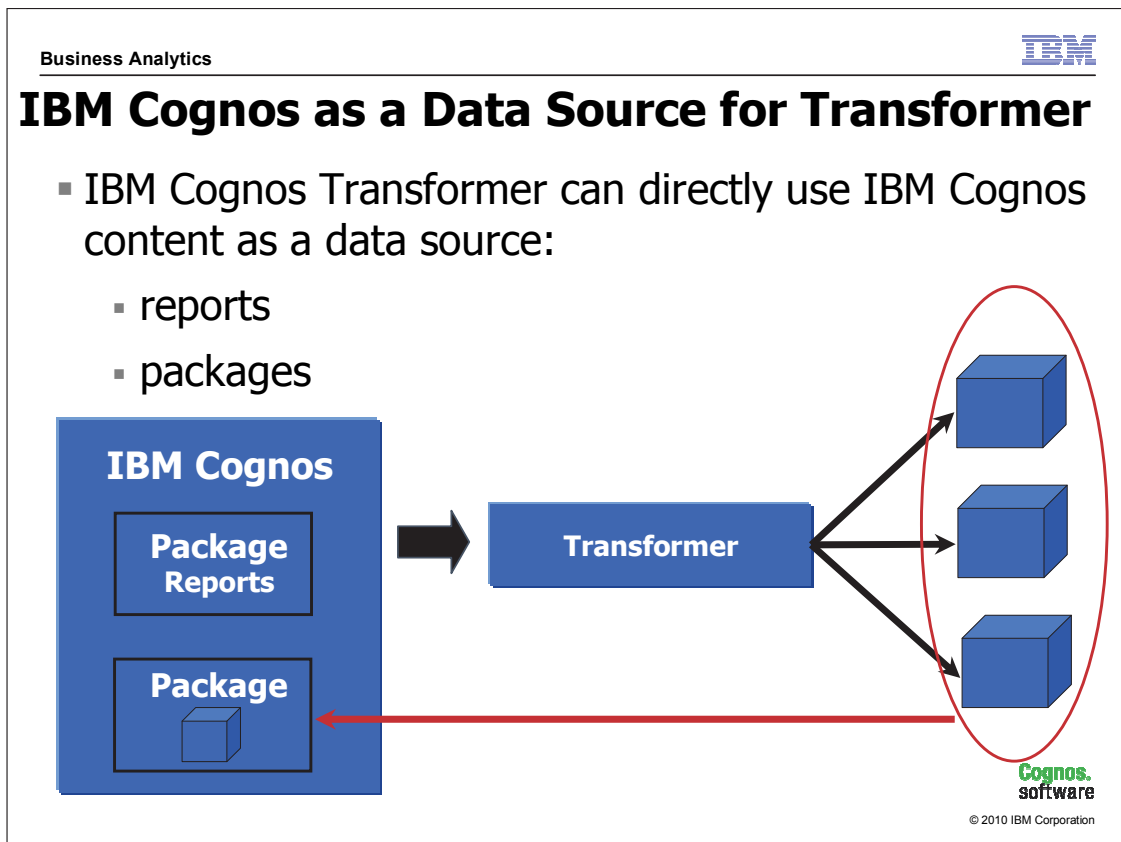
5. Close **Notepad**, and then examine the data in the **Business view_Time dimension.tab**.

This file also contains column headers, and all the data from the time dimension.

6. Close **Notepad**.
7. In **Framework Manager**, save the project.

Results:

You externalized query subjects from your model in tab delimited file format so that the data retrieved can be leveraged in other applications.

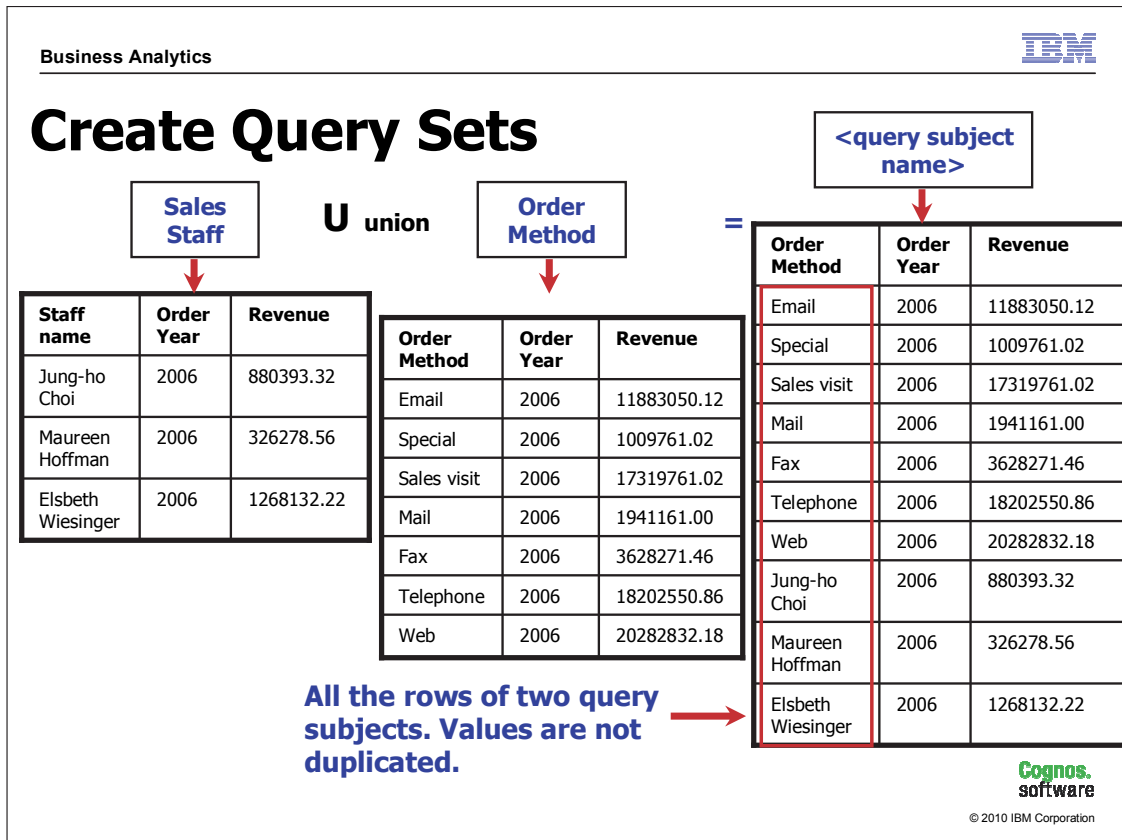


Transformer is the IBM Cognos OLAP modeling component used to model and build PowerCubes.

As a modeler, you can make metadata available for use in Transformer. For data to be used in dimensions, you need to make items available in your dimension query subjects to act as category codes (business keys) and captions. The business key values should be unique across all levels in the dimension. For fact data, you need to make items available in your fact query subjects that Transformer will use as measures and as the keys that allow the measures to be related to the dimensions. For example, if your measures will relate to the Product dimension, you will need to make the Product Key available in your fact query subject.

Once the appropriate metadata is made available, Transformer modelers can access the individual dimension information and measure information (each accessed as a separate data source) through IBM Cognos reports, or directly from metadata in the package.

Once a Transformer modeler has created a PowerCube, it can be published as a package in IBM Cognos for use as a multidimensional data source.



You define a query set to merge, compare, or equate similar data from different sources. Query sets are useful when modeling data from disparate systems, or when the desired result is not directly available through a typical query of the data source.

A query set can be defined using the set operations of union, intersect, or except.

Demo 3: Create a Query Set

Purpose:

Report authors must provide reports that include data values from both the Staff name and Order method query items in a single column, to allow easier comparison. They can achieve this result in Report Studio for a given report; however, they must repeat this operation for multiple reports that share this requirement. They would like to avoid this, and have this functionality included as part of the model. To meet this requirement, you will create a query set using a union operation that will combine the values from these two query items into a single column.

Component: Framework Manager

Project: great_outdoors_sales

Task 1. Create the first query subject.

1. In Framework Manager, in the **Project Viewer**, under **go_sales>Database View>gosales>Model query subjects (gosales)**, select the following items:

Query Subject	Query Item
Order method	Order method
Sales	Order date Revenue

2. Right-click the **Order method** query item, and then click **Merge in New Query Subject**.
3. When prompted to create relationships, click **No**, rename the new query subject to **Order Methods Revenue**.

Instead of reporting against order dates, authors would like the comparison by year. You will use the Year function to extract the year from Order date.

4. Expand **Order Methods Revenue**, double-click **Order date**, and then edit the **Expression definition** so that it appears as follows:

Year([gosales].[Sales].[Order date])

5. Click **OK**, rename **Order date** to **Order year**.

For testing purposes, you will create a filter that will narrow the scope of the result set to the year 2006.

6. Double-click **Order Methods Revenue**, click the **Filters** tab, in the bottom right corner, click **Add**, and then in the **Name** box, type **Order Year Filter**.
7. In the **Available Components** pane, double-click **Order year** to add it to the **Expression definition** pane.
8. At the end of the expression, type **=2006**,

The results appear as follows:

[gosales].[Order Methods Revenue].[Order year] = 2006

9. Click **OK**, and then click **OK** again.

Task 2. Create the second query subject.

1. In the **Project Viewer**, expand **gohr>Model query subjects (gohr)**.
2. Select the following items from both the gosales and gohr namespaces:

Query Subject	Query Item
Sales staff	Staff name
Sales	Order date Revenue

3. Right-click **Staff name**, click **Merge in New Query Subject**, and then click **No** to creating relationships.
4. Rename the new query subject to **Sales Staff Revenue**, move it to the **Model query subjects (gohr)** folder, and then expand the **Sales Staff Revenue** query subject.
5. Double-click **Order date**, and then edit the **Expression definition** so that it appears as follows:
Year([gosales].[Sales].[Order date])
6. Click **OK**, rename **Order date** to **Order year**, and then click **Enter**.
Again, for testing purposes, you will create a filter that will narrow the scope of the result set to the year 2006.
7. Double click **Sales Staff Revenue**, click the **Filters** tab, in the bottom right corner, click **Add**, and then in the **Name** box, type **Order Year Filter**.
8. Double-click **Order year** to add it to the **Expression definition** pane.
9. At the end of the expression, type **=2006**,
The results appear as follows:
[gohr].[Sales Staff Revenue].[Order year] = 2006
10. Click **OK**, and then click **OK** again.

Task 3. Create the query set.

1. Click **Order Methods Revenue**, and then Ctrl+click **Sales Staff Revenue**.
2. From the **Actions** menu, click **Define Query Set**.
3. In the **Name** box, type **Order Methods/Sales Staff Revenue for 2006**.

The default set operation is to Union the two query subjects. This is the selection you want.

4. Click **OK**, and then expand the **Order Methods/Sales Staff Revenue for 2006** query subject.

There are only 3 query items in the query subject. By default, Framework Manager has changed the Usage property of the Order year query item to Fact. You will change this.

5. Under **Order Methods/Sales Staff Revenue for 2006**, click **Order year**, and then in the **Properties** pane, change the **Usage** property to **Attribute**.
6. Test the **Order Methods/Sales Staff Revenue for 2006** query subject.

The results appear as follows:

Test result		
Order method	Order year	Revenue
Web	2006	18560
Web	2006	69258
Nathalie Benoit	2006	11203.4
Abram Ruiz	2006	6182.85
James Ross-Hythe	2006	4546.04
Web	2006	15917.55
Web	2006	6999.52
Wilbur Baldock	2006	8694.4
Mathilde Leuder	2006	3994

Notice that the Order method column contains values based on both the original Order method and Staff name query subjects. This is a result of the Union operation performed. You will now rename the query item appropriately and implement a technique that will sort the values in the column.

Task 4. Rename query item and add a sort key query item to the original query subjects.

1. Click **Close**, and then rename the **Order method** query item to **Order methods/Staff names**.
2. Double-click the **Order Methods Revenue** query subject, and then, in the bottom right corner, click **Add**.
3. In the **Name** box, type **Sort Key**, in the **Expression definition** box, type '**A**', and then click **OK**.
4. Click **OK** again.
5. Repeat steps **2** to **4** to create a query item called **Sort Key** in the **Sales Staff Revenue** query subject that has the following expression: '**B**'.

6. Double-click the **Order Methods/Sales Staff Revenue for 2006** query subject, and then click **OK**.

The Sort Key query item appears in the Order Methods/Sales Staff Revenue for 2006 query subject.

7. Test the **Order Methods/Sales Staff Revenue for 2006** with **Auto Sum** enabled.

The results appear as follows:

Test results				
	Order methods/Staff names	Order year	Revenue	Sort Key
	Aaltje Hansen	2006	8503567.85	B
	Abram Ruiz	2006	10450641.98	B
	Adda Heijman	2006	8641892.44	B
	Adriaantje Haanraads	2006	10210608.27	B
	Agatha Reyes	2006	7378224.86	B
	Agnelo Chavez	2006	8132618	B
	Agnes Ramos	2006	7355606.6	B
	Aidan Chaplin	2006	4873573.41	B
	Aiko Watanabe	2006	13789738.8	B
	Aila Forssell	2006	11234064.27	B
	Aimi Tanaka	2006	3525878.21	B
	Akemi Takahashi	2006	16197823.54	B
	Akemi Yamada	2006	13167128.19	B
	Akira Hashimoto	2006	1181719.82	B
	Alberto Pera	2006	6529169.45	B
	Alessandra Torta	2006	8881435.6	B
	Alexandre Pereira	2006	10870383.67	B

The Order methods/Staff names column appears with values sorted based on the presence of the Sort Key column.

8. Click **OK**, close **Framework Manager**, and then click **Yes** to save changes.

Results:

You created a query set that unions the values from the Staff name and Order method query items into a single column.

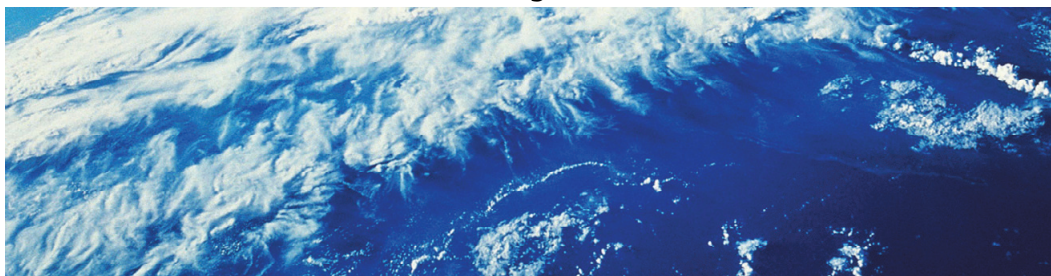
Summary

- You should now be able to:
 - leverage a user defined function
 - set the order of operations in a model calculation
 - externalize query subjects
 - prepare IBM Cognos content for use as a data source in Transformer
 - create query sets



Model Multilingual Metadata

IBM Cognos BI

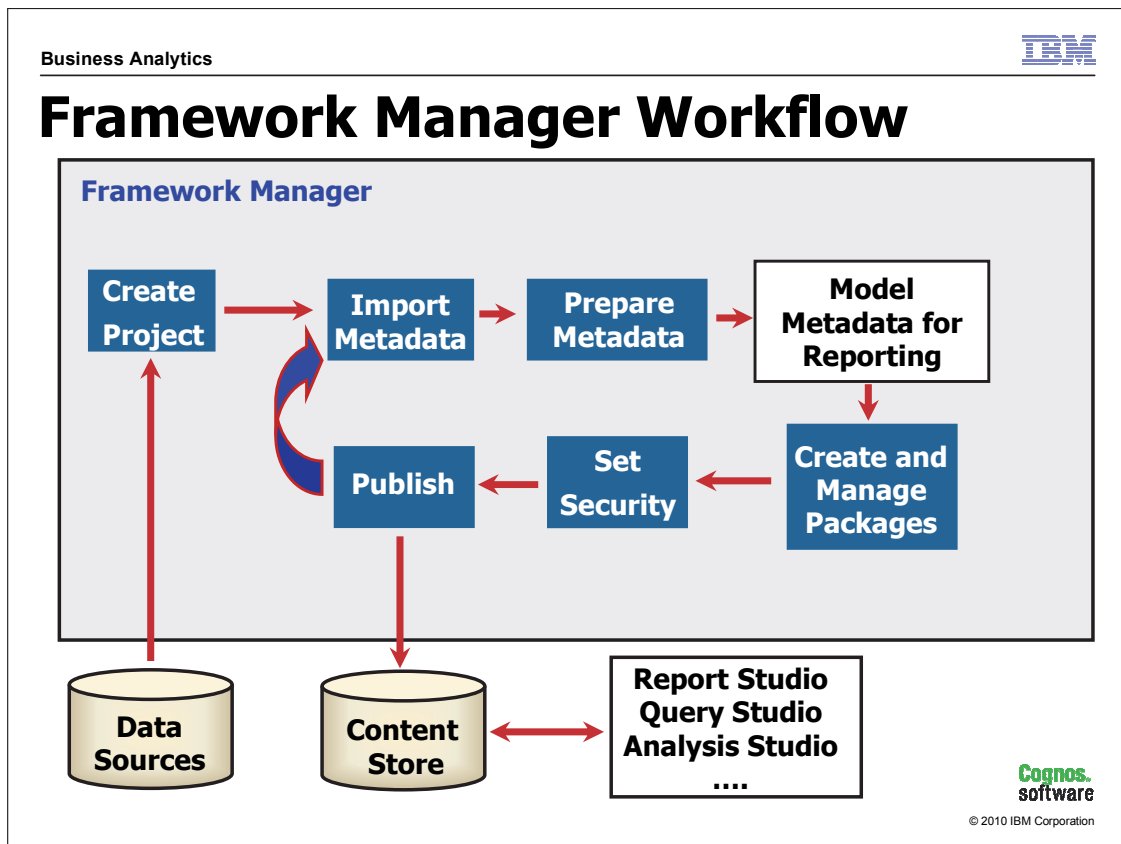


Business Analytics

© 2010 IBM Corporation

Objectives

- At the end of this module, you should be able to:
 - customize metadata for a multilingual audience



This module describes modifying language properties to ensure that metadata is available in all languages required for authors.

Modify Language Properties

- Each model object has three Language properties:
 - Name
 - Description
 - Screen Tip
- Language properties can be modified:
 - manually in the project
 - by exporting/importing a translation file

You can translate the multilingual text properties associated with metadata; names, descriptions, and screen tips. Only the metadata is translated. You can then publish your package to make the translated metadata available to authors. To do this:

- define the languages to be included in the project
- modify the Language properties manually or export the multilingual text properties to a translation file, which translators use to input the correct text for each language. Once translation is complete, import the file to update the Language properties with the translated strings.
- specify the languages you want to include in your package (they must be available at the project level first) and then publish the metadata.

Demo 1: Apply Multilingual Query Item Properties

Purpose:

You want to ensure that users working in languages other than English can easily use the GO Operational model to create reports. You will add French support to the project, translate a query item name and description, publish the package with both French and English metadata, and test the package in Report Studio.


Component: Framework Manager

Project: GO Operational

Task 1. Add languages to the project.

1. In **Framework Manager**, close any projects that may be open, and then open the **GO Operational** project located at **C:\Edcognos\B5152\CBIFM-Start Files\Appendix B\GO Operational**.

This project has had any additional languages removed from the model in order to perform this demonstration.

2. From the **Project** menu, point to **Languages**, and then click **Define Languages**.
3. In the **Available languages** box, click **French**, click **Add** , and then click **OK**.


A warning message appears indicating that the languages will be added to every text property of every object.

4. Click **OK**.

Task 2. Add a French name and description.

1. Expand **GO Operational Model>Consolidation View>Products**, and then click the **Product Type** query item.
2. At the top of the **Properties** window, click the **Language** tab.

The results appear as follows:

Properties						
Properties		Language				
	Name		Description		Screen Tip	
	English ▼	French ▼	English ▼	French ▼	English ▼	French ▼
 Product Type	Product Type	(fr) Product T...				

Notice that for the Name, Description and Screen Tip properties, you now see one entry for each language. However, all entries are in English initially, the project's design language.

The description and screen tip are blank for the French columns. You cannot add a French value for either the description or screen tip until you supply a value for the primary language (English).

3. In the **Description** row under **English**, type **Product Type identifies a set of related products**, and then press **Enter**.

You can now add a French value, because you have supplied a value for the primary language.

4. In the **Description** row under **French**, delete the existing text, type **Type de produit** and then press **Enter**.
5. In the **Name** row under **French**, type **Type de produit**.
6. Save the project.

Task 3. Publish the model in both English and French, and create a report.

1. Under **Packages**, select the **GO Operational** package, and then, from the **Actions** menu, click **Package>Specify Package Languages**.
2. Under **Available Project Languages**, select **French**, click **Add**, and then click **OK**.
3. Publish the **GO Operational** package.

Note: In the Properties pane for packages, you can also translate the package name and any descriptions or tool tips required by end users.

4. Launch **IBM Cognos Connection**, log on, and then open **Report Studio** selecting the **GO Operational** package.
5. Click **Create new**, and then double-click **List**.
6. In the **Insertable Objects** pane, expand **Presentation View>GO Operational Sales (query)>Sales (query)>Products**.
7. Right-click **Product Type**, and then click **Properties**.

In the Description row of the Properties dialog box, the text appears that you specified for the Description property in Task 2.

8. Click **Close**, and then close **Report Studio** without saving.

Task 4. Create a report to verify the French query item name and description.

1. In **IBM Cognos Connection**, click **My Home**, click **My Area Options** in the top right corner, and then click **My Preferences**.
2. Under **Regional options**, change the **Content language** setting to **Use the following language**, in the list click **French**, and then click **OK**.

3. Launch **Report Studio** selecting the **(fr) GO Operational** package.
4. Create a new **List** report.

Report Studio opens with the metadata from the (fr) GO Operational model appearing in the Insertable Objects pane. Notice that all items are prefixed with (fr). This indicates that you are now in the French version of the model but you have not yet translated all of your metadata titles and properties.

5. Expand **(fr) Presentation View > (fr) GO Operational Sales (query) > (fr) Sales (query) > (fr) Products**.

Notice that Type de produit stands out among the (fr)-prefixed query item names.

6. Right-click **Type de Produit**, and then click **Properties**.

In the Description row of the Properties dialog box, the text appears that you specified for the Description property in Task 2.

7. Click **Close**, and then close **Report Studio**.
8. In **IBM Cognos Connection**, click **My Area Options** in the top right corner, and then click **My Preferences**.
9. Under **Regional options** and **Content language**, click **Use the default language**, and then click **OK**.

Results:

You added French support to the project, translated a query item name and description, published the package in both French and English, and tested the package in Report Studio.

Demo 2: Apply Multilingual Translation Files



Purpose:

You want to ensure that users working in languages other than English can easily use the GO Operational model to create reports. To this end, you will enhance the model by exporting a translation file and modifying it so that it contains French and English strings. You will then import this file back into the model and view the results.

Component: Framework Manager

Project: GO Operational


Task 1. Export metadata to a CSV file for translation.

1. In **Framework Manager**, from the **Project** menu, point to **Languages**, and then click **Export Translation File**.
2. Under the **Project Languages** pane, Ctrl+click **French** so that both **English** and **French** are selected, and then click **Add**  to move them to the **Languages to be exported** pane.
3. Next to the **Export languages to this file** box, click **Browse** .
4. In the **Save as type** box, click **CSV (comma delimited) (*.csv)**.
5. Click **Desktop** in the left pane, in the **File name** box, type **GO_Application_LOC.csv**, and then click **Save**.
6. Click **OK**.

A message appears, indicating that the language strings were successfully exported.

7. Click **OK**.
8. On the **Windows Desktop**, double-click **GO_Application_LOC.csv** to open it in **Microsoft Excel**, and then expand the first two columns.
Notice that each column represents a given language; in this case, English and French. These are based on the language selections you made when you exported the model languages in previous steps.
9. In the second column of row **18**, change the **French** value of **(fr) Branch City** to **Ville de Succursale**.
10. In the second column of row **21**, change the **French** value of **(fr) Branch Country** to **Pays de Succursale**.
11. Save the file, click **Yes** to the warning message and close **Excel**.

Task 2. Import a CSV file that contains translated strings.

1. In **Framework Manager**, from the **Project** menu, point to **Languages**, and then click **Import Translation File**.
2. In the **Project Languages** pane, click **French**, and then, below **Translate into**, click **Add**  to add it to the **Translate into** pane.
3. Next to the **Import translation table from this file** box, click **Browse**.
4. In the **Files of type** box, click **CSV (comma delimited) (*.csv)**, and then, if necessary, click **Desktop**, and then double-click **GO_Application_LOC.csv**.
5. Click **OK**.

A message appears indicating that the import was successful and the details of the properties that were updated in the model objects. All objects that reference the translated metadata are updated such as items in the Foundation Objects View, the Consolidation View and the Dimensional View.

6. Click **OK**.
7. In the **Consolidation View**, expand **Branch by Location**, and then click **Branch Country**.

In the Properties pane, notice that the French value for the Name property of Branch Country reflects the change that you made to the translation file.

8. Click **Branch City**.

Again, in the Properties pane, notice that the French value for the Name property of the query item is as expected.

9. Save and close the project.

Results:

You enhanced the GO Operational model by exporting the model languages to a translation file. You modified this file so that it contained French and English strings. You imported the file back into the model and viewed the results. Users can now begin to work with the GO Operational model in languages other than English.

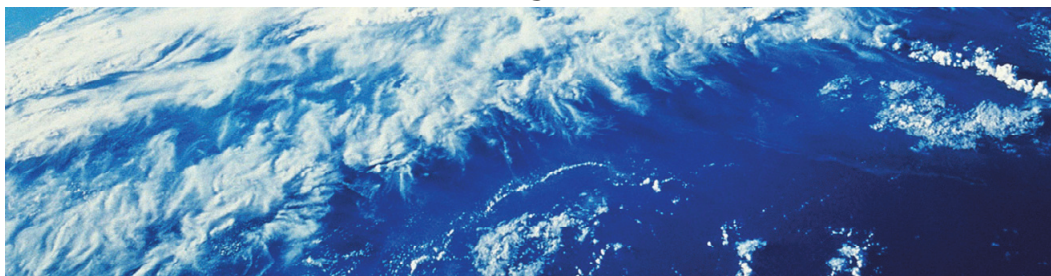
Summary

- You should now be able to:
 - customize metadata for a multilingual audience



Additional Resources

IBM Cognos BI



Business Analytics

© 2010 IBM Corporation

IBM Cognos provides a resource for users and developers to guide them through the tasks and processes of maximizing the IBM Cognos business intelligence tools. This resource is the Cognos Proven Practices documentation repository.

Created by Cognos experts from real-life customer experiences, Cognos Proven Practices is your source for rich technical information that is tried, tested, and proven to help you succeed with Cognos products in your specific technology environment.

The Cognos Proven Practices documentation repository is updated regularly. Access the repository through IBM Developerworks. The Cognos specific URL is:

<http://www.ibm.com/developerworks/data/library/cognos/cognosprovenpractices.html>

In particular, the following topics and documents augment this course:

- Time Period Analysis

<http://www.ibm.com/developerworks/data/library/cognos/page352.html>

This document describes a technique for Dimensionally Modeled Relational data sources that will allow for relative time period analysis within crosstabs. This technique focuses on developing period-to-date aggregates.

- Durable Models

<http://www.ibm.com/developerworks/data/library/cognos/page60.html>

This document will point out some concepts which will help you to create models that will be as flexible as possible and help survive changes to the requirements.

<http://www.ibm.com/developerworks/data/library/cognos/modeling/design/page496.html>

This document describes a process to ensure that changes to a Framework Manager model's published packages and corresponding content in the Content Store will continue to function in unison even as changes are made to the model.

Check this valuable resource regularly.

